Temă - Laborator03

Petculescu Mihai-Silviu

Enunț

Să se realizeze o aplicație pentru administrarea contactelor, cu posibilitatea de a `adauga`, `sterge`, `modifica`, `cauta` contacte.

Un contact are următoarea structură:

ContactModel

```
class ContactModel {
  public int Id { get; set; }
  public string Name { get; set; }
  public string Email { get; set; }
  public string Phone { get; set; }
  public string Address { get; set; }
  public string Country { get; set; }
  public override bool Equals(object obj) {
    if (obj == null) {
      return false;
    }
    if (! (obj is ContactModel)) {
      return false;
    }
    ContactModel c = obj as ContactModel;
    return Id == c.Id && Name == c.Name && Email == c.Email && Phone == c.Phone;
  }
}
```

Se vor realiza 2 proiecte:

- `Contact.Models` de tip library `.dll`
- `Contact.UI` de tip `Windows Forms`

În `Contact.Models` se vor implementa 2 clase: `ContactModel` și `ContactList`

ContactList

```
public class ContactList {
  public List < ContactModel > Contacts { get; set; }
  public ContactList() {
    Contacts = new List < ContactModel > ();
  }
  public void Add(ContactModel contact) {
    Contacts.Add(contact);
  }
  public void Remove(ContactModel contact) {
    Contacts.Remove(contact);
  }
  public bool SearchByName(string name) {
    foreach(var contact in Contacts) {
      if (contact.Name == name)
        return true;
    }
    return false;
  }
  public bool SearchByEmail(string email) {
    // De Implementat
  }
  public void SaveOnDisk() {
    // Salvati toate contactele intr-un fisier pe disk!
  }
  public void LoadFromDisk() {
    // Colectia "Contacts" va fi incarcata cu contactele salvate pe disk prin
metoda anterioara
  }
}
```

În al doilea proiect va fi partea de interfață:

- o fereastră care permite:
  - adaugare
  - modificare
  - stergere
  - cautare (email sau nume)
- de contacte și vizualizarea unei liste de contacte (un obiect de tip `ContactList`).

Temă

App.config

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <startup>
        <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2" />
    </startup>
    <appSettings>
        <add key="DiskPath" value="D:\ContactList.txt" />
    </appSettings>
</configuration>
```

Content.UI.Form1

```
using Contact.Models;
using System;
using System.Collections.Generic;
using System.Windows.Forms;

namespace Contact.UI {
  public partial class Form1: Form {

    ContactList contactList;
    private int initialIndex;

    public Form1() {
      InitializeComponent();
      cb_Sort.Items.Add("Name");
      cb_Sort.Items.Add("Email");
      cb_Sort.SelectedIndex = 0;
    }

    private void Form1_Load(object sender, EventArgs e) {
      contactList = new ContactList();
      contactList.LoadFromDisk();
      LoadSourceInDataGrid(contactList.Contacts);
    }

    private void bSave_Click(object sender, EventArgs e) {
      contactList.SaveOnDisk();
    }

    private void LoadSourceInDataGrid(List < ContactModel > dataSource) {
      BindingSource source = new BindingSource();
      source.DataSource = dataSource;
      dataGridView.AutoGenerateColumns = true;
      dataGridView.AutoSizeColumnsMode =
DataGridViewAutoSizeColumnsMode.DisplayedCells;
      dataGridView.DataSource = source;
    }

    private void bLoad_Click(object sender, EventArgs e) {
      contactList.LoadFromDisk();
      LoadSourceInDataGrid(contactList.Contacts);
    }

    private void dataGridView_CellEndEdit(object sender,
DataGridViewCellEventArgs e) {
      var index = e.RowIndex;
      if (initialIndex == -1) return;
      ContactModel model = (ContactModel)
dataGridView.Rows[index].DataBoundItem;
      contactList.UpdateByIndex(initialIndex, model);
    }

    private void bSort_Click(object sender, EventArgs e) {
      string text = tbSort.Text;
      if (cb_Sort.SelectedItem.ToString() == "Name")
LoadSourceInDataGrid(contactList.FilterByName(text));
```

```
      else if (cb_Sort.SelectedItem.ToString() == "Email")
LoadSourceInDataGrid(contactList.FilterByEmail(text));
    }

    private void bRefresh_Click(object sender, EventArgs e) {
      LoadSourceInDataGrid(contactList.Contacts);
    }

    private void dataGridView_CellBeginEdit(object sender,
DataGridViewCellCancelEventArgs e) {
      var index = e.RowIndex;
      initialIndex = contactList.GetIndex((ContactModel)
dataGridView.Rows[index].DataBoundItem);
    }

    private void bClean_Click(object sender, EventArgs e) {
      contactList.CleanContacts();
      LoadSourceInDataGrid(contactList.Contacts);
    }
  }
}
```

Contact.Model.ContactModel

```
namespace Contact.Models {
  public class ContactModel {
    public int Id { get; set; }
    public string Name { get; set; }
    public string Email { get; set; }
    public string Phone { get; set; }
    public string Address { get; set; }
    public string Country { get; set; }
    public ContactModel() {}
    public ContactModel(string[] args) {
      Id = int.Parse(args[0]);
      Name = args[1];
      Email = args[2];
      Phone = args[3];
      Address = args[4];
      Country = args[5];
    }
    public override bool Equals(object obj) {
      if (obj == null || !(obj is ContactModel))
        return false;
      ContactModel c = obj as ContactModel;
      return Id == c.Id && Name == c.Name && Email == c.Email && Phone ==
c.Phone;
    }
    public override string ToString() {
      return $ "{Id}: {Name}, {Email}, {Phone}, {Address}, {Country}";
    }
    public override int GetHashCode() {
      return base.GetHashCode();
    }
  }
}
```

Contact.Model.ContactList

```
using System;
using System.Collections.Generic;
using System.Configuration;
using System.IO;
using System.Linq;

namespace Contact.Models {
  public class ContactList {
    public List < ContactModel > Contacts { get; set; }

    public ContactList() {
      Contacts = new List < ContactModel > ();
    }

    public void Add(ContactModel contact) {
      Contacts.Add(contact);
    }

    public void Remove(ContactModel contact) {
      Contacts.Remove(contact);
    }

    public void UpdateByIndex(int index, ContactModel contact) {
      if (!Contacts[index].Equals(contact)) Contacts[index] = contact;
    }

    public bool SearchByName(string name) {
      foreach(var contact in Contacts)
      if (contact.Name == name) return true;
      return false;
    }

    public bool SearchByEmail(string email) {
      foreach(var contact in Contacts)
      if (contact.Email == email) return true;
      return false;
    }

    public List < ContactModel > FilterByName(string name) {
      List < ContactModel > sorted = new List < ContactModel > ();
      if (SearchByName(name) == true) foreach(var contact in Contacts) if
(contact.Name == name) sorted.Add(contact);
      return sorted;
    }

    public List < ContactModel > FilterByEmail(string email) {
      List < ContactModel > sorted = new List < ContactModel > ();
      if (SearchByEmail(email) == true) foreach(var contact in Contacts) if
(contact.Email == email) sorted.Add(contact);
      return sorted;
    }

    // Salvati toate contactele intr-un fisier pe disk!
    public void SaveOnDisk() {
      var filePath = ConfigurationManager.AppSettings["DiskPath"];
      File.WriteAllText(filePath, "");
      foreach(ContactModel contact in Contacts)
      File.AppendAllText(filePath, $ "{contact}\n");
    }

    // Colectia "Contacts" va fi incarcata cu contactele salvate pe disk prin
metoda anterioara
    public void LoadFromDisk() {
      var filePath = ConfigurationManager.AppSettings["DiskPath"];
      CleanContacts();
      List < string > lines;
      try {
        lines = File.ReadLines(filePath).ToList();
      }
      catch(Exception) {
        File.WriteAllText(filePath, "");
        lines = File.ReadLines(filePath).ToList();
      }
      foreach(var contact in lines) {
        var fields = contact.Split(new char[] { ':', ',' }).Select(o
=>o.Trim()).ToArray();
        ContactModel model = new ContactModel(fields);
        Contacts.Add(model);
      }
    }

    public void CleanContacts() {
      Contacts.Clear();
    }

    public int GetIndex(ContactModel contact) {
      for (int i = 0; i < Contacts.Count; i++)
      if (Contacts[i].Equals(contact)) return i;
      return - 1;
    }
  }
}
```