Laborator04

Utilizarea librariilor externe in aplicatiile Visual Studio

Exemplu: serializarea/deserializarea obiectelor C# in format Json si salvarea acestora pe disk / recuperarea acestora de pe disk.

JSon (Javascript Object Notation)

Format de tip text ce permite reprezentarea obiectelor ca string, in structuri ierarhice.

Aplicatie: sa se realizeze o clasa Product, o clasa Order si o clasa Basket (cos de cumparaturi) ce poate sa contina mai multe comenzi (obiecte Order).

Sa se implementeze posibilitatea salvarii/recuperarii pe/de pe disk in forma JSon, utilizand o librarie externa (de exemplu "Newtonsoft")

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Newtonsoft.Json;

namespace BasketModels
{
    //salveaza / citeste obiecte de diverse tipuri (Product, Order, Basket) de pe
disk
    public class DataManagement
    {
        public static void Save(Product product)
        {
            if (product == null || string.IsNullOrEmpty(product.BarCode))
            {
                return;
            }

            //salvez in format JSon fisierul pe disk,
            //cu numele identic cu product.BarCode

            //serializez ca json obiectul si salvez pe disk
            var jsonString = JsonConvert.SerializeObject(product);

            File.WriteAllText($"Product_{product.BarCode}.json", jsonString);
        }

        public static void Save(Order order)
        {
            if (order == null || order.Id == 0)
            {
                return;
            }

            //salvez in format JSon fisierul pe disk,
```

```
            //cu numele identic cu order.Id

            //serializez ca json obiectul si salvez pe disk
            var jsonString = JsonConvert.SerializeObject(order);

            File.WriteAllText($"Order_{order.Id}.json", jsonString);
        }

        public static void Save(Basket basket)
        {
            if (basket == null)
            {
                return;
            }

            //salvez in format JSon fisierul pe disk,

            //serializez ca json obiectul si salvez pe disk
            var jsonString = JsonConvert.SerializeObject(basket);

            File.AppendAllText($"Basket.json", jsonString);
        }

        public static Product GetProduct(string barCode)
        {
            if (string.IsNullOrEmpty(barCode))
            {
                return null;
            }

            //verificam daca exista pe disk fisierul cu acest barcode!
            var fileName = $"Product_{barCode}.json";

            if (!new FileInfo(fileName).Exists)
            {
                return null;
            }

            var jsonValue = File.ReadAllText(fileName);

            return JsonConvert.DeserializeObject<Product>(jsonValue);
        }

        public static Order GetOrder(int orderId)
        {

            //verificam daca exista pe disk fisierul cu acest orderId!
            var fileName = $"Order_{orderId}.json";

            if (!new FileInfo(fileName).Exists)
            {
                return null;
            }

            var jsonValue = File.ReadAllText(fileName);

            return JsonConvert.DeserializeObject<Order>(jsonValue);
        }
```

```
        public static Basket GetBasket()
        {
            //verificam daca exista pe disk fisierul cu acest orderId!
            var fileName = $"Basket.json";

            if (!new FileInfo(fileName).Exists)
            {
                return null;
            }

            var jsonValue = File.ReadAllText(fileName);

            return JsonConvert.DeserializeObject<Basket>(jsonValue);
        }

    }
}
```

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

namespace BasketModels
{
    public class Basket
    {
        public List<Order> Orders { get; private set; }

        public Basket()
        {
            Orders = new List<Order>();
        }

        public void AddOrder(Order order)
        {
            Orders.Add(order);
        }

        public void Remove(Order order)
        {
            Orders.Remove(order);
        }

        public double GetTotal()
        {
            double result = 0;

            foreach(Order order in Orders)
            {
                var product = DataManagement.GetProduct(order.ProductBarcode);

                if (product != null)
                {
                    result += product.Price * order.Count;
```

```
                }
            }

            return result;
        }
    }
}
```

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

namespace BasketModels
{
    public class Product
    {
        public string BarCode { get; set; }
        public double Price { get; set; }
        public string Name { get; set; }
        public string Category { get; set; }
        public DateTime CreateDate { get; set; }
        public DateTime ExpirationDate { get; set; }
    }
}
```

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

namespace BasketModels
{
    public class Order
    {
        public int Id { get; set; }
        public DateTime Date { get; set; }
        public string ProductBarcode { get; set; }
        public int Count { get; set; }

        public override bool Equals(object obj)
        {
            if (obj == null)
            {
                return false;
            }

            if (!(obj is Order))
            {
                return false;
            }

            return Id == (obj as Order).Id;
        }
```

```
    }
}
```

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

namespace BasketApplication
{
    class Program
    {
        static void Main(string[] args)
        {
            Basket basket = new Basket();
            Product p1 = new Product()
            {
                BarCode = "1001a",
                CreateDate = DateTime.Now,
                ExpirationDate = DateTime.Now.AddYears(3),
                Category = "electronics",
                Name = "iphone",
                Price = 2000
            };

            Product p2 = new Product()
            {
                BarCode = "1002b",
                CreateDate = DateTime.Now,
                ExpirationDate = DateTime.Now.AddYears(10),
                Category = "electronics",
                Name = "tv",
                Price = 2500
            };

            DataManagement.Save(p1);
            DataManagement.Save(p2);

            Order o1 = new Order()
            {
                Id = 1,
                ProductBarcode = "1001a",
                Date = DateTime.Now,
                Count = 3
            };

            Order o2 = new Order()
            {
                Id = 2,
                ProductBarcode = "1002b",
                Date = DateTime.Now,
                Count = 5
            };

            DataManagement.Save(o1);
```

```
            DataManagement.Save(o2);

            basket.AddOrder(o1);
            basket.AddOrder(o2);

            Console.WriteLine("Totals: {0}", basket.GetTotal());

            DataManagement.Save(basket);

            Console.ReadKey();
        }
    }
}
```

```
// get all products
public static List<Product> GetAllProducts()
        {
            var result = new List<Product>();

            //cautam pe disk toate fisierele care incep cu "Product_";
            var files = Directory.GetFiles(".", "Product_*.json");

            foreach(var file in files)
            {
                var jsonValue = File.ReadAllText(file);

                result.Add(JsonConvert.DeserializeObject<Product>(jsonValue));
            }

            return result;
        }
```

Tema Laborator04:

Sa se adauge o interfata grafica (UI) de tip Windows Forms la aplicatia anterioara care sa permita urmatoarele:

1. Adaugare produse(Product);
2. Creare comenzi (Order) si adaugare in cos de cumparaturi (Basket)

Observatie: interesant ar fi sa permiteti utilizatorului sa selecteze doar din lista de produse deja create (existente pe disk in urma activitatii de la 1)

Cum s-ar implementa??

Utilizati metoda GetAllProducts si aflati toate codurile de bare pe care le puteti afisa intr-o lista de timp "combobox"; de acolo poate selecta user-ul codul de bare dorit

3. Salvare produse,comenzi,basket pe disk
4. Modificarea (editarea) datelor despre produse / comenzi (salvarea pe disk a modificarilor)
5. vizualizarea continutului cosului de cumparaturi (se vor inclusiv informatii despre produsele comandate, nu doar codul de bare!!)
6. eliminare comenzi din cos
7. calcul cost total comenzi din cos

Timp de lucru: 1 week.