Gdynia, 23.02.2022

PROGRAMOWANIE .NET

LABORATORIUM NR 1

Serializacja/Deserializacja

Autor:

Grzegorz Konopka, nr albumu: 59388

1. Cel laboratorium

Celem było zapoznanie się z podstawami tworzenia aplikacji standalone w środowisku .NET przy użyciu Windows Form Application. Podczas realizacji zadania zapoznaliśmy się z procesem serializacji i deserializacji danych. Na potrzeby przedstawienia procesu deserializacji i serializacji wybrałem format JSON (JavaScript Object Notation).

2. Serializacja/Deserializacja – definicje

Serializacja – w programowaniu proces przekształcania obiektów, tj. instancji określonych klas, do postaci szeregowe do strumienia danych z zachowaniem aktualnego stanu obiektu.

Deserializacja – polega na odczytaniu wcześniej zapisanego strumienia danych i odtworzeniu na tej podstawie obiektu klasy wraz z jego stanem bezpośrednio sprzed serializacji.

Popularne formaty serializacji:

- Strumień bajtów
- XML
- YAML
- JSON

3. Zastosowanie serializacij

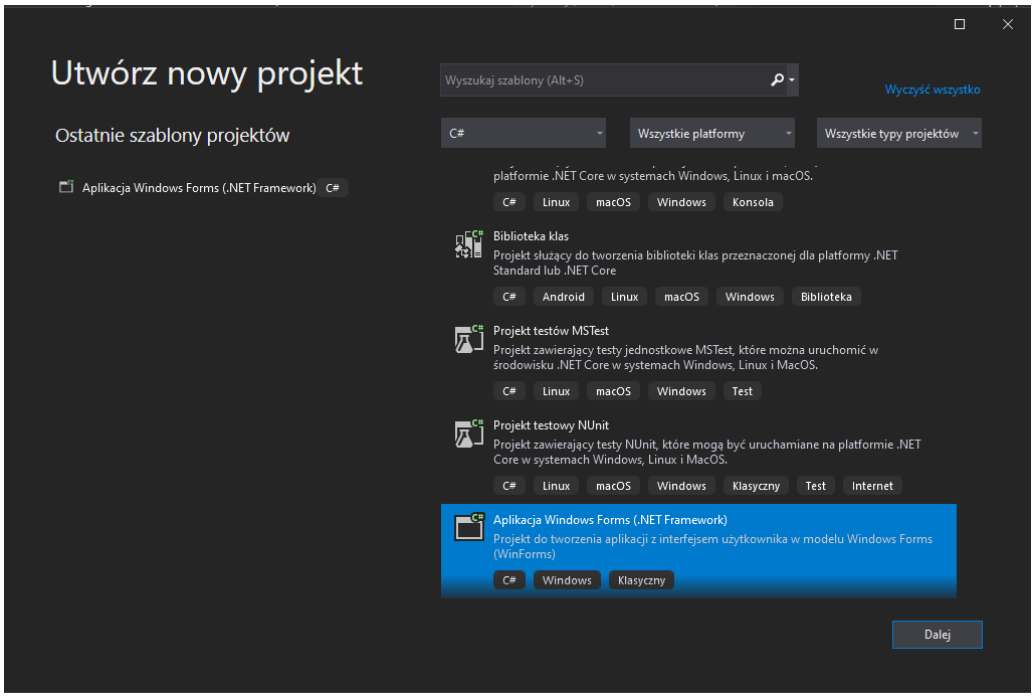
Proces serializacji i deserializacji jest szeroko wykorzystywany we współczesnych aplikacjach webowych – najpopularniejszym formatem jest JSON. Dzięki takiemu podejściu projektowana aplikacja nie jest w zasadzie zależna od technologii – można część frontendową stworzyć np. w ReactJS, a backendową w Javie, a nie jak to miało miejsce w przeszłości np. backend i frontend w Javie (przy użyciu JSF, JSP lub Thymeleaf).

4. Treść ćwiczenia

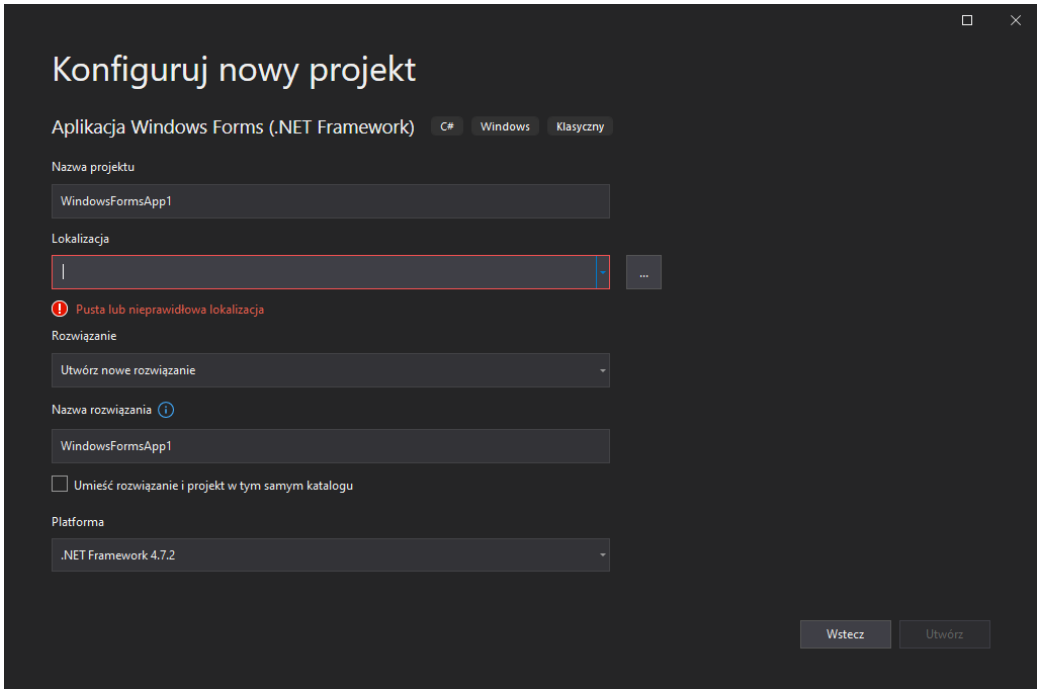
Zaimplementować w języku C# przy użyciu Windows Forms Application aplikację, która demonstruje zasadę działania serializacji i deserializacji obiektów.

5. Wykonanie zadania

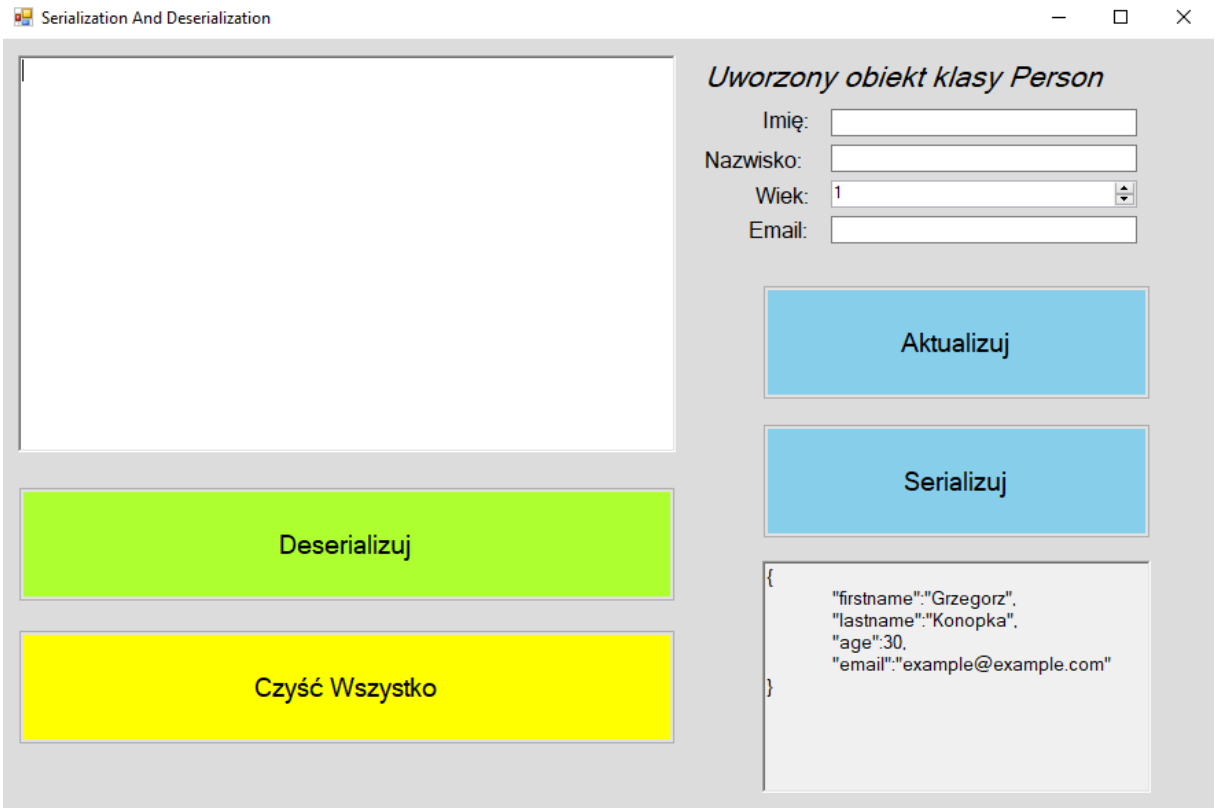
5.1. Utwórz nowy projekt w Windows Application Forms dla platformy .NET lub .NET Core i kliknij Dalej:



5.2. Nadaj nazwę projektu i wskaż lokalizację projektu. Kliknij utwórz:



5.3. W kreatorze GUI stwórz formę o następującym wyglądzie:



5.4. Nadaj wszystkim obiektom nazwę i właściwości zgodne z tabelą poniżej:

NR	TYP OBIEKTU	NAZWA	WŁAŚCIWOŚCI	AKCJA
1.	Okno	mainForm	Nazwa: Serialization And Deserialization	-
2.	RichTextBox	textArea		-
3.	Button	deserializeButton	Kolor: zielony Text: Deserializuj	deserializeButton_Click(object sender, EventArgs e)
4.	Button	clearButton	Kolor: żółty Text: Czyść Wszystko	clearAllButton_Click(object sender, EventArgs e)
5.	Button	updateButton	Kolor: niebieski Text: Aktualizuj	updateButton_Click(object sender, EventArgs e)
6.	Button	serializeButton	Kolor: niebieski Text: Serializuj	serializeButton_Click(object sender, EventArgs e)
7.	RichTextBox	exampleArea	ReadyOnly: True	-
8.	Label	personInstanceLabel	Text: Utworzony obiekt klasy Person	-
9.	Label	firstnameLabel	Text: Imię	-
10.	Label	lastnameLabel	Text: Nazwisko	-
11.	Label	ageLabel	Text: Wiek	-
12.	Label	emailLabel	Text: Email	-
13.	TextBox	firstnameText		-
14.	TextBox	lastnameText		-

15.	NumericUpDown	agePicker	Minimum: 1 Maximum: 100	-
16.	TextBox	emailText		-

5.5. Zaimportuj z NuGet pakiet Newtonsoft.Json.

5.6. Stwórz klasę Person.cs zgodnie z listingiem poniżej:

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

namespace SerializationAndDeserialization
{
    class Person
    {
        [JsonProperty]
        private string firstname;

        public Person(String firstname, String lastname, int age,
String email)
        {
            this.firstname = firstname;
            this.lastname = lastname;
            this.age = age;
            this.email = email;
        }

        public string GetFirstname()
        {
            return firstname;
        }

        public void SetFirstname(string value)
        {
            firstname = value;
        }
        [JsonProperty]
        private string lastname;

        public string GetLastname()
        {
            return lastname;
        }

        public void SetLastname(string value)
        {
            lastname = value;
        }
        [JsonProperty]
        private int age;

        public int GetAge()
```

```
        {
            return age;
        }

        public void SetAge(int value)
        {
            age = value;
        }
        [JsonProperty]
        private string email;

        public string GetEmail()
        {
            return email;
        }

        public void SetEmail(string value)
        {
            email = value;
        }
    }
}
```

5.7. Stwórz klasę PersonMapper.cs która będzie odpowiedzialna za mapowanie instancji klasy Person.cs zgodnie z listingiem poniżej:

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

namespace SerializationAndDeserialization
{
    static class PersonMapper
    {
        public static Person deserialize(String json)
        {
            Person person =
JsonConvert.DeserializeObject<Person>(json);
            return person;
        }

        public static String serialize(Person person)
        {
            String json = JsonConvert.SerializeObject(person);
            return json;
        }
    }
}
```

5.8. W klasie Form1.cs zaimplementuj:

5.8.1. Metodę dla obiektu nr 3:

```
      private void deserializeButton_Click(object sender, EventArgs
e)
```

```
        {
            try
            {
                person = PersonMapper.deserialize(textArea.Text);
                firstnameText.Text = person.GetFirstname();
                lastnameText.Text = person.GetLastname();
                agePicker.Value = person.GetAge();
                emailText.Text = person.GetEmail();
            } catch (Exception exception)
            {
                textArea.Clear();
                textArea.Text = "Nie można zdeserializować
obiektu!";
            }
        }
```

5.8.2. Metodę dla obiektu nr 4:

```
        private void clearAllButton_Click(object sender, EventArgs
e)
        {
            firstnameText.Clear();
            lastnameText.Clear();
            agePicker.Value = agePicker.Minimum;
            emailText.Clear();
            textArea.Clear();
        }
```

5.8.3. Metodę dla obiektu nr 5:

```
        private void updateButton_Click(object sender, EventArgs e)
        {
            if (firstnameText.Text.Trim().Length > 0
                && lastnameText.Text.Trim().Length > 0
                && emailText.Text.Trim().Length > 0)
            {
                person.SetFirstname(firstnameText.Text);
                person.SetLastname(lastnameText.Text);
                person.SetAge(Convert.ToInt32(agePicker.Value));
                person.SetEmail(emailText.Text);
            }
        }
```

5.8.4. Metodę dla obiektu nr 6

```
        private void serializeButton_Click(object sender, EventArgs
e)
        {
            textArea.Text = PersonMapper.serialize(person);
        }
```

5.8.5. Zainicializuj obiekt klasy Person w klasie Form1.cs:

```
        Person person = null;
```

5.9. Doprowadź klase Form1.cs do stanu zgodnego z poniższym:

```
using Newtonsoft.Json;
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 SerializationAndDeserialization
{
    public partial class mainForm : Form
    {

        Person person = null;

        public mainForm()
        {
            InitializeComponent();
        }

        private void deserializeButton_Click(object sender,
EventArgs e)
        {
            try
            {
                person = PersonMapper.deserialize(textArea.Text);
                firstnameText.Text = person.GetFirstname();
                lastnameText.Text = person.GetLastname();
                agePicker.Value = person.GetAge();
                emailText.Text = person.GetEmail();
            } catch (Exception exception)
            {
                textArea.Clear();
                textArea.Text = "Nie można zdeserializować
obiektu!";
            }
        }

        private void clearAllButton_Click(object sender, EventArgs
e)
        {
            firstnameText.Clear();
            lastnameText.Clear();
            agePicker.Value = agePicker.Minimum;
            emailText.Clear();
            textArea.Clear();
        }
```

```
        private void updateButton_Click(object sender, EventArgs e)
        {
            if (firstnameText.Text.Trim().Length > 0
                && lastnameText.Text.Trim().Length > 0
                && emailText.Text.Trim().Length > 0)
            {
                person.SetFirstname(firstnameText.Text);
                person.SetLastname(lastnameText.Text);
                person.SetAge(Convert.ToInt32(agePicker.Value));
                person.SetEmail(emailText.Text);
            }
        }

        private void serializeButton_Click(object sender, EventArgs
e)
        {
            textArea.Text = PersonMapper.serialize(person);
        }
    }
}
```

5.10. Bibliografia

5.10.1. https://en.wikipedia.org/wiki/Serialization

5.10.2. https://www.newtonsoft.com/json/help/html/SerializingJSON.htm

5.10.3. https://docs.microsoft.com/pl-pl/dotnet/csharp/programming-guide/concepts/serialization/