Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»
Филиал
«Минский радиотехнический колледж»

Учебный предмет
«Конструирование программ и языки программирования»

Инструкция
по выполнению лабораторной работы №21-22
«Разработка, отладка и испытание программ записи и чтения данных в json-файл»

Минск 2024 г.

Лабораторная работа № 21-22

Тема работы: «Разработка, отладка и испытание программ чтения и записи данных в json-файл»

1 Цель работы

Сформировать умение разрабатывать программы чтения и записи данных в json-файл.

2 Задание

Организовать чтение и запись данных в json-файл для заданий из лабораторной работы №19 согласно Вашего варианта.

3 Оснащение работы

ПК, среда Visual Studio 2019, MSword.

4 Основные теоретические сведения

Основная функциональность по работе с JSON сосредоточена в пространстве имен **System.Text.Json**.

Ключевым типом является класс **JsonSerializer**, который и позволяет сериализовать объект в json и, наоборот, десериализовать код json в объект C#.

Для сохранения объекта в json в классе JsonSerializer определен статический метод **Serialize()**, который имеет ряд перегруженных версий. Некоторые из них:

- `string Serialize(Object obj, Type type, JsonSerializerOptions options)`: сериализует объект obj типа type и возвращает код json в виде строки. Последний необязательный параметр options позволяет задать дополнительные опции сериализации
- `string Serialize<T>(T obj, JsonSerializerOptions options)`: типизированная версия сериализует объект obj типа T и возвращает код json в виде строки.
- `Task SerializeAsync(Object obj, Type type, JsonSerializerOptions options)`: сериализует объект obj типа type и возвращает код json в виде строки. Последний необязательный параметр options позволяет задать дополнительные опции сериализации
- `Task SerializeAsync<T>(T obj, JsonSerializerOptions options)`: типизированная версия сериализует объект obj типа T и возвращает код json в виде строки.
- `object Deserialize(string json, Type type, JsonSerializerOptions options)`: десериализует строку json в объект типа type и возвращает десериализованный объект. Последний необязательный параметр options позволяет задать дополнительные опции десериализации

- `T Deserialize<T>(string json, JsonSerializerOptions options)`: десериализует строку json в объект типа T и возвращает его.
- `ValueTask<object> DeserializeAsync(Stream utf8Json, Type type, JsonSerializerOptions options, CancellationToken token)`: десериализует текст UTF-8, который представляет объект JSON, в объект типа type. Последние два параметра необязательны: options позволяет задать дополнительные опции десериализации, а token устанавливает CancellationToken для отмены задачи. Возвращается десериализованный объект, обернутый в ValueTask
- `ValueTask<T> DeserializeAsync<T>(Stream utf8Json, JsonSerializerOptions options, CancellationToken token)`: десериализует текст UTF-8, который представляет объект JSON, в объект типа T. Возвращается десериализованный объект, обернутый в ValueTask

Рассмотрим применение класса на простом примере. Сериализуем и десериализуем простейший объект:

```
using System;
using System.Text.Json;
namespace HelloApp
{
   class Person
   {
      public string Name { get; set; }
      public int Age { get; set; }
   }
   class Program
   {
      static void Main(string[] args)
      {
         Person tom = new Person { Name = "Tom", Age = 35 };
         string json = JsonSerializer.Serialize<Person>(tom);
         Console.WriteLine(json);
         Person restoredPerson = JsonSerializer.Deserialize<Person>(json);
         Console.WriteLine(restoredPerson.Name);
      }
   }
}
```

Здесь вначале сериализуем с помощью метода `JsonSerializer.Serialize()` объект типа Person в стоку с кодом json. Затем обратно получаем из этой строки объект Person посредством метода `JsonSerializer.Deserialize()`.

Консольный вывод:

```
{"Name":"Tom","Age": 35}
```

```
Tom
```

Хотя в примере выше сериализовался/десериализовался объект класса, но подобным способом мы также можем сериализовать/десериализовать структуры.

Некоторые замечания по сериализации/десериализации

Объект, который подвергается десериализации, должен иметь конструктор без параметров. Например, в примере выше этот конструктор по умолчанию. Но можно также явным образом определить подобный конструктор в классе.

Сериализации подлежат только публичные свойства объекта (с модификатором public).

Запись и чтение файла json

Поскольку методы SerializeAsyc/DeserializeAsync могут принимать поток типа Stream, то соответственно мы можем использовать файловый поток для сохранения и последующего извлечения данных:

```
using System;
using System.IO;
using System.Text.Json;
using System.Threading.Tasks;

namespace HelloApp
{
    class Person
    {
        public string Name { get; set; }
        public int Age { get; set; }
    }
    class Program
    {
        static async Task Main(string[] args)
        {
            // сохранение данных
            using (FileStream fs = new FileStream("user.json", FileMode.OpenOrCreate))
            {
                Person tom = new Person() { Name = "Tom", Age = 35 };
                await JsonSerializer.SerializeAsync<Person>(fs, tom);
                Console.WriteLine("Data has been saved to file");
            }

            // чтение данных
            using (FileStream fs = new FileStream("user.json", FileMode.OpenOrCreate))
            {
```

```
            Person restoredPerson = await JsonSerializer.DeserializeAsync<Person>(fs);
            Console.WriteLine($"Name: {restoredPerson.Name} Age: {restoredPerson.Age}");
        }
    }
  }
}
```

В данном случае вначале данные сохраняются в файл user.json и затем считываются из него.

5. Порядок выполнения работы

1. Выделить ключевые моменты задачи.
2. Построить алгоритм и теоретическую объектную модель решения задачи.
3. Запрограммировать полученные алгоритмы и объектную модель.

6. Форма отчета о работе

Лабораторная работа № ___

Номер учебной группы _____

Фамилия, инициалы учащегося _____

Дата выполнения работы _____

Тема работы: _____

Цель работы: _____

Оснащение работы: _____

Результат выполнения работы: _____

7. Контрольные вопросы и задания

1. Для сохранения объекта в json в классе JsonSerializer определен …
2. Поясните понятие «Сериализация/десереализация»
3. Может ли объект, который подвергается десериализации, иметь конструктор без параметров?
4. Каким образом осуществляется запись файла json?
5. Каким образом осуществляется чтение файла json?

8. Рекомендуемая литература

1. Рихтер, Дж. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C# / Дж. Рихтер. СПб. : Изд-во Питер, 2021. 896 с.

2. Прайс, М. Дж. C# 10 и .NET 6. Современная кросс-платформенная разработка / М. Дж. Прайс. СПб : Изд-во Питер, 2023. 848 с.

3. Васильев, А.Н. Программирование на C# для начинающих. Особенности языка / А.Н. Васильев. М. : Эксмо, 2022. 528 с.

4. Фримен, А. ASP.NET Core 3 с примерами на C# для профессионалов / А. Фримен. СПб. : Изд-во Вильямс, 2021. 1184 с.