Практическая работа №14
Сериализация и десериализация данных в формате JSON

1Цель работы

1.1 Научиться выполнять сериализацию и десериализацию данных в формате JSON в приложениях на C#.

2 Литература

2.1 https://docs.microsoft.com/ru-ru/dotnet/standard/serialization/system-text-json-how-to?pivots=dotnet-5-0

2.2 https://metanit.com/sharp/tutorial/6.5.php

3 За дание

3.1 Сериализация объектов класса с полями простых типов данных.

3.1.1 Создать класс для хранения имени и возраста человека.

Для хранения данных в классе вместо полей использовать открытые автореализуемые свойства:

public тип ИмяСвойства { get; set; }

3.1.2 Создать в основной программе:

- объект созданного класса, вызвав конструктор по умолчанию,
- объект созданного класса, присвоив значения свойствам,
- массив объектов созданного класса.

Для инициализации свойств при создании объекта можно использовать следующий код:

```
ИмяКласса переменая = new ИмяКласса
{
        Свойство1 = значение1,
        Свойство2 = значение2
};
```

3.1.3 Используя метод JsonSerializer.Serialize(данные), выполнить сериализацию информации о созданных объектах и массиве. Результат вывести на экран.

3.1.4 Сохранить в отдельных файлах формата .json сериализованные данные.

Для записи данных в файл можно использовать следующий метод:

File.WriteAllText(имя файла, json строка);

3.2 Применение опций при сериализации данных

Указать вторым параметром метода объект типа JsonSerializerOptions со свойствами, отвечающими за:

- игнорирование null-значений;
- приведение к стилю CamelCase
- форматирование.

Сравнить результаты, полученные с использованием этих опций и без них.

3.3 Десериализация объектов класса с полями простых типов данных.

Используя метод JsonSerializer.Deserialize<ТипДанных>(json строка), считать данные из сохраненных файлов в отдельные переменные типа var. Результат десериализации проверить в режиме отладки.

3.4 Настройка атрибутов сериализации

3.4.1 Создать класс для хранения информации о пользователе. Добавить в класс:

- открытые автореализуемые свойства идентификатор, логин и список комментариев (List<string>),
- открытое поле типа, созданного в п.3.1, для возможности указания имени и возраста пользователя.

3.4.2 Создать в основной программе:

- объект созданного класса, вызвав конструктор по умолчанию,
- объект созданного класса, присвоив значения всем свойствам,
- объект созданного класса, присвоив значения всем свойствам и полю.

3.4.3 Используя метод JsonSerializer.Serialize(данные), выполнить сериализацию информации о созданных объектах и списке комментариев одного из объектов. При сериализации выполнять форматирование json-файла. Результат вывести на экран.

3.4.4 Добавить следующие атрибуты в созданном в п.3.4.1 классе:

- у поля атрибут, отвечающий за сериализацию поля,
- у идентификатора атрибут, пропускающий это свойство при сериализации
- атрибут, изменяющий имя json-свойства список комментариев на отличающийся от исходного названия свойства.

Сравнить результат сериализации, полученный после задания атрибутов, с исходным.

3.5 Десериализация объектов класса с полями ссылочных типов данных.

Выполнить десериализацию объектов из п.3.4. Результат десериализации проверить в режиме отладки.

4 Порядок выполнения работы

4.1 Выполнить все задания из п.3 в решении PractWork14. Каждый класс должен быть в отдельном файле. Возможные ошибки требуется обрабатывать. Выполнить форматирование и рефакторинг кода.

4.2 Ответить на контрольные вопросы.

5 Содержание отчета

5.1 Титульный лист

5.2 Цель работы

5.3 Ответы на контрольные вопросы

5.4 Вывод

6 Контрольные вопросы

6.1 Что такое «JSON» и для чего применяется этот формат?

6.2 Что такое «сериализация»?

6.3 Что такое «десериализация »?

6.4 Какое пространство имен нужно подключить для обработки данных в формате JSON?

6.5 Какова общая форма вызова метода сериализации?

6.6 Какова общая форма вызова метода десериализации?

6.7 Какое пространство имен нужно подключить для указания атрибутов сериализации в формате JSON?