**Практическая работа №4**

**Оглавление**

[Практическая работа № 4. Работа с json файлами 3](#_Toc163812729)

[Содержание пояснительной записки 12](#_Toc163812730)

[Используемое программное обеспечение 13](#_Toc163812731)

[Список литературы 14](#_Toc163812732)

Практическая работа № 4.  
Работа с json файлами

**Цель практической работы**

Закрепление теоретических знаний по работе с данными в формате JSON.

**Постановка задачи**

**Сериализация** — это процесс **преобразования объекта** в поток байтов (в нашем случае **в строку**) для сохранения или передачи в память, базу данных или файл. Эта операция предназначена для того, чтобы **сохранить состояния объекта** для последующего воссоздания при необходимости. Обратный процесс называется **десериализацией**.

Например, мы хотим создать объект типа Employee и сохранить его в файл, чтобы потом можно было загрузить с заполненными данными.

public class Employee

{

public string FirstName { get; set; } = "";

public string LastName { get; set; } = "";

public double Salary { get; set; }

public Employee(

string firstName,

string lastName,

double salary)

{

FirstName = firstName;

LastName = lastName;

Salary = salary;

}

}

Для начала нужно скачать с помощью NuGet пакетов библиотеку **Newtonsoft.JSON** (рисунок 1).

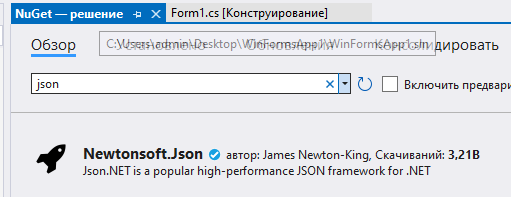


Рисунок 1 – Установка библиотеки Newtonsoft.Json

Создаем объект сотрудника, и сериализуем в строку формата JSON, далее сохраняем в файл стандартным способом:

var employee = new Employee("John", "Black", 3000);

var jsonString = JsonConvert.SerializeObject(employee);

File.WriteAllText("file.json", jsonString);

В данном случае **сериализуются все публичные поля и свойства!**

Полученный файл представлен на рисунке 2. Данный файл можно открыть любым текстовым редактором, также удобно использовать **VisualStudio** или **Notepad++**.

Данные по сотруднику также можно было сохранить в виде текста вручную: сформировать самим строку определенного формата и сохранить ее, однако тогда нужно было бы продумать обратную операцию получения объекта из строки. Очевидно, что гораздо удобнее пользоваться готовым решением в виде JSON.

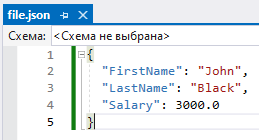


Рисунок 2 – Содержимое JSON файла

**Для десериализации** (получения объекта из JSON строки) нужно написать:

var employee1 = JsonConvert.DeserializeObject<Employee>(jsonString);

Далее рассмотрим, **что делать, если нужно сериализовать приватное поле или свойство, или наоборот если что-то сериализовать не нужно.**

Атрибуты в .NET представляют специальные инструменты, которые позволяют встраивать в сборку дополнительные **метаданные**.

**Атрибуты указываются в квадратных скобках**, над классом, полем или свойством.

Чтобы явно указать, **что объект можно сериализовать** используется атрибут [Serializable] (для рассматриваемой библиотеки он не обязателен).

Если мы хотим задавать свои имена у данных **в файле JSON** нужно использовать.

[DataContract] перед именем класса, а сами поля и свойства пометить [DataMember].

Чтобы отметить нужный конструктор для десериализации, нужно использовать атрибут [JsonConstructor], при этом имена переменных в конструкторе должны совпадать с именами соответствующих полей и свойств (без учета регистра).

[DataContract]

public class Employee

{

[DataMember(Name = "Имя")]

public string FirstName { get; set; } = "";

[DataMember(Name = "Фамилия")]

public string LastName { get; set; } = "";

[DataMember(Name = "Зарплата")]

public double Salary { get; set; }

[JsonConstructor]

public Employee(string firstName, string lastName, double salary)

{

FirstName = firstName;

LastName = lastName;

Salary = salary;

}

}

На рисунке 3 представлен результат сериализации объекта с заданными именами полей.

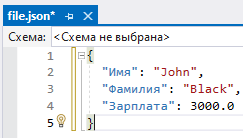


Рисунок 3 – Содержимое JSON файла  
с заданными именами полей

Чтобы указать, что поле не нужно сериализовать необходимо отметить его атрибутом [JsonIgnore], результат представлен на рисунке 4.

[DataContract]

public class Employee

{

[JsonIgnore]

public string FirstName { get; set; } = "";

[DataMember(Name = "Фамилия")]

public string LastName { get; set; } = "";

[DataMember(Name = "Зарплата")]

public double Salary { get; set; }

[JsonConstructor]

public Employee(string firstName, string lastName, double salary)

{

FirstName = firstName;

LastName = lastName;

Salary = salary;

}

}

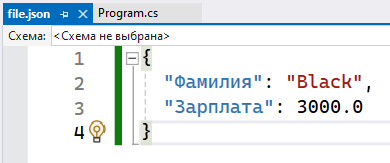


Рисунок 4 – Содержимое JSON файла  
с исключением поля «FirstName»

Также можно сериализовать вложенные классы. Пример сериализации объекта класса, содержащего внутри себя другие классы:

[DataContract]

public class Company

{

[DataMember(Name = "Директор")]

public Employee Director { get; set; }

[DataMember(Name = "Сотрудники")]

public List<Employee> Employees { get; set; } = new();

[JsonConstructor]

public Company(Employee director, List<Employee> employees)

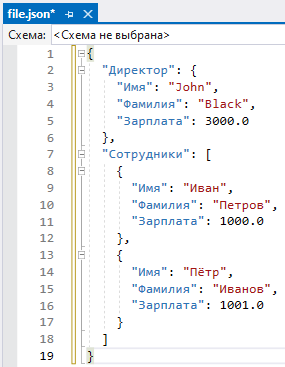
{

Director = director;

Employees = employees;

}

}



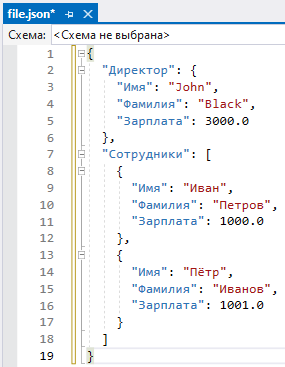


Рисунок 5 – Содержимое JSON файла с вложенными классами

Если объекты имеют сложную и запутанную структуру, являются наследниками класса, требуют вызова каких-либо метода для инициализации, то автоматическая сериализация будет затруднена или невозможна. Рассмотрим пример «ручной» сериализации.

var employee = new Employee("John", "Black", 3000);

JObject j = new JObject();

j["Имя"] = employee.FirstName;

j["Фамилия"] = employee.LastName;

j["Зарплата"] = employee.Salary;

var status = employee.Salary switch

{

< 100 => "Бедняк",

< 1000 => "Средний класс",

< 5000 => "Богач",

\_ => "Сказочный богач"

};

j["Статус"] = status;

File.WriteAllText("file.json", j.ToString());

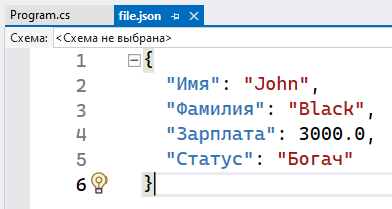


Рисунок 6 – Содержимое JSON файла после ручного заполнения

Видно, что работа с JSON напоминает работу со словарем. напишем код для чтения данного файла.

var j\_str = File.ReadAllText("file.json");

var json = JObject.Parse(j\_str);

var employee1 = new Employee(

json["Имя"].Value<string>(),

json["Фамилия"].Value<string>(),

json["Зарплата"].Value<int>());

**Задание на практическую работу**

1. Создать Windows Forms приложение, реализовать графический интерфейс для создания, сохранения и загрузки **объекта** согласно варианту.
2. При загрузке объекта его параметры записываются в интерфейс формы.
3. Поля и свойства **объекта** придумать самим, у **объекта** должно быть не менее **4** свойств или полей. Для сохранения и загрузки использовать **JSON**.
4. Защита работы включает демонстрацию работы программы на подготовленном примере.

**Варианты заданий на практическую работу**

Вариант 1. Продуктовый магазин

Вариант 2. Банковский счет

Вариант 3. Студент

Вариант 4. Завод

Вариант 5. Смартфон

Вариант 6. Компьютер

Вариант 7. Продукт (в магазине)

Вариант 8. Грузовой автомобиль

Вариант 9. Книжный шкаф

Вариант 10. Портфель

Вариант 11. Банк

Вариант 12. Жилой дом

Вариант 13. Ракета

Вариант 14. Офицер

Вариант 15. Порт (для кораблей)

Вариант 16. Животное

Вариант 17. Страна

Вариант 18. Самолет

Вариант 19. Материал

Вариант 20. Планета

Вариант 21. Адресная книга

Вариант 22. Аквариум

Вариант 23. Департамент

Вариант 24. Игра

Вариант 25. Артиллерийское орудие

Вариант 26. Сундук

Вариант 27. Армия

Вариант 28. Огород

Вариант 29. Танк

Вариант 30. Корабль

# Содержание пояснительной записки

1. Постановка задачи. Приводится теоретический материал, использованный при написании приложения.

2. Формулировка задания и вариант. Приводится задание на лабораторную работу и вариант этого задания.

3. Описание выполняемых действий. Необходимо привести описание последовательности разработки программы, реализации используемых методов, алгоритмов, блок-схем.

4. Анализ результатов. Привести анализ входных и выходных данных. Показать результаты выполнения программного кода. Предоставить скриншоты обработки тестовых примеров. Сделать выводы.

5. Листинг программы. Привести листинг разработанного программного кода, содержание файлов входных и выходных данных.

# Используемое программное обеспечение

1. Среда программирования MS Visual Studio Community 2022 (Свободно распространяемое программное обеспечение (в учебных целях));
2. Microsoft Office Standard 2007 (Open License: 42267924);
3. Open Office (Свободно распространяемое программное обеспечение).
4. Браузер (Свободно распространяемое программное обеспечение).

# Список литературы

* + - 1. Мейер Б. Объектно-ориентированное программирование и программная инженерия [Электронный ресурс]/ Мейер Б. – Электрон. текстовые данные. – М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. – 285 c.
      2. Биллиг, В. A. Основы объектного программирования на С# (C# 3.0, Visual Studio 2008) [Электронный ресурс] : учебное пособие / В. A. Биллиг. — Электрон. текстовые данные. — Москва, Саратов : Интернет-Университет Информационных Технологий (ИНТУИТ), Вузовское образование, 2017. — 583 c. — 978-5-4487-0145-0. — Режим доступа: http://www.iprbookshop.ru/72339.html
      3. Павловская, Т. А. Программирование на языке высокого уровня C# [Электронный ресурс] / Т. А. Павловская. — 2-е изд. — Электрон. текстовые данные. — М. : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. — 245 c. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/73713.html
      4. Агапов, В. П. Основы программирования на языке С# [Электронный ресурс] : учебное пособие / В. П. Агапов. — Электрон. текстовые данные. — М. : Московский государственный строительный университет, ЭБС АСВ, 2012. — 128 c. — 978-5-7264-0576-6. — Режим доступа: http://www.iprbookshop.ru/16366.html
      5. Медведев, М. А. Программирование на СИ# [Электронный ресурс] : учебное пособие / М. А. Медведев, А. Н. Медведев ; под ред. А. В. Присяжный. — Электрон. текстовые данные. — Екатеринбург : Уральский федеральный университет, ЭБС АСВ, 2015. — 64 c. — 978-5-7996-1561-1. — Режим доступа: http://www.iprbookshop.ru/69667.html
      6. Казанский А.А. Объектно-ориентированное программирование на языке Microsoft Visual С# в среде разработки Microsoft Visual Studio 2008 и .NET Framework. 4.3 [Электронный ресурс]: учебное пособие и практикум/ Казанский А.А.— Электрон. текстовые данные.— М.: Московский государственный строительный университет, ЭБС АСВ, 2011.— 180 c
      7. Уйманова Н.А. Основы объектно-ориентированного программирования [Электронный ресурс]: практикум/ Уйманова Н.А., Таспаева М.Г.— Электрон. текстовые данные.— Оренбург: Оренбургский государственный университет, ЭБС АСВ, 2017.— 156 c.— Режим доступа: http://www.iprbookshop.ru/78808.html.— ЭБС «IPRbooks»
      8. Новиков П.В. Объектно-ориентированное программирование [Электронный ресурс]: учебно-методическое пособие к лабораторным работам/ Новиков П.В.— Электрон. текстовые данные.— Саратов: Вузовское образование, 2017.— 124 c.— Режим доступа: http://www.iprbookshop.ru/64650.html.— ЭБС «IPRbooks»