Практическая работа №16
Сериализация и десериализация данных в формате CSV

1Цель работы

1.1 Научиться выполнять сериализацию и десериализацию данных в формате CSV в приложениях на C#, используя механизм рефлексии.

2 Литература

2.1 https://docs.microsoft.com/ru-ru/dotnet/framework/reflection-and-codedom/reflection
2.2 https://metanit.com/sharp/tutorial/14.1.php

3 За дание

3.1 Получение списка открытых свойств класса

3.1.1 Создать класс для хранения данных пользователя (идентификатор, логин, пароль, возраст) и класс для хранения данных о человеке (имя, возраст).

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

public тип ИмяСвойства { get; set; }

3.1.2 Создать в основной программе обобщенный метод:

string GetHeaders<T>(T object)

Метод предназначен для формирования строки на основе названий открытых свойств класса по следующему шаблону (количество свойств должно совпадать с количеством открытых свойств класса):

Свойство1;Свойство2;Свойство3;

Для получения списка свойств используется механизм рефлексии:

```
Type currentType = typeof(T); // получение описания типа объекта
var properties = currentType.GetProperties(); // получение списка свойств типа
```

Имена свойств получаются перебором списка свойств (имя текущего свойства – свойство Name текущего свойства).

3.1.3 Проверить работу метода, вызвав его для объектов класса Человек и Пользователь.

3.2 Формирование списка данных на основе открытых свойств класса

3.2.1 Создать в основной программе обобщенный метод:

string Serialize<T>(List<T> values)

Метод предназначен для сериализации данных в csv-формат.

Описание формата csv:

- первая строка – строка заголовков, заголовки разделяются точкой с запятой
- каждая следующая строка – описание значений свойств определенного объекта, значения также разделяются точкой с запятой.

Для хранения значений свойств объектов создать массив rows (размер массива совпадает с количеством элементов в values**)**.

Для получения значения свойства объекта по его имени использовать следующий код:

// переменная, в которую будет записано значение свойства
object currentValue = null;
// чтение значения свойства объекта по имени свойства
currentValue = currentType.GetProperty(свойство.Name).GetValue(объект);

Если значение свойства currentValue равно null, то вместо него указывается пустая строка, иначе – само значение, приведенное к строковому виду. После каждого значения – точка с запятой.

После заполнения всего массива выполнить объединение элементов массива, например, используя String.Join()

3.2.2 Проверить работу метода, вызвав его для списков объектов класса Человек и Пользователь.

3.3 Экранирование строковых данных (заключение строк в кавычки)

3.3.1 Модифицировать метод из п.3.2 следующим образом: значения свойств типа String должны заключаться в двойные кавычки.

Тип данных свойства определяется следующим образом:

свойство.PropertyType.Name

3.3.2 Проверить работу метода, вызвав его для списков объектов класса Человек и Пользователь.

3.4 Создание списка объектов на основе данных в формате csv

3.4.1 Создать в основной программе обобщенный метод:

List<T> Deserialize<T>(string values)

Метод предназначен для десериализации данных из csv-формата.

Реализовать создание в методе списка объектов типа T, количество элементов списка должно быть получено из данных в values (можно их разбить методом Split).

Для создания объекта использовать следующий код:

T объект = (T)Activator.CreateInstance(currentType);

Значения свойств в этом задании не учитываются (только количество объектов)

3.4.2 Проверить работу метода, вызвав его для списков объектов класса Человек и Пользователь.

3.5 Чтение списка данных на основе открытых свойств класса

3.5.1 Модифицировать метод из п.3.4, чтобы при создании объектов заполнялись значения их свойств:

PropertyInfo свойство = obj.GetType().GetProperty(имя свойства);

свойство.SetValue(объект, Convert.ChangeType(значение, свойство.PropertyType));

3.5.2 Проверить работу метода, вызвав его для списков объектов класса Человек и Пользователь.

4 Порядок выполнения работы

4.1 Выполнить все задания из п.3 в решении PractWork16. 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 csv-формат?
6.2 Что такое «рефлексия» в программировании?
6.3 Как получить описание типа данных в C#?
6.4 Как получить список открытых свойств типа данных в C#?
6.5 Каков синтаксис создания объекта при использовании класса Activator?