Основы программирования

Средства ввода-вывода

Сериализация

Бинарная сериализация

Для бинарной сериализации в .NET можно применять класс `BinaryFormatter`.

Отметим, что хотя в принципе его можно применять, но не рекомендуется, поскольку он не считается безопасным.

Данная часть приведена только для справки.

Сериализация и десериализация

При работе с байтовыми потоками состояние объекта пишется в поток по-элементно, с последующим поэлементным же считыванием.

В некоторых ситуациях удобно записывать в поток объект целиком.

Процесс преобразования объекта в линейный поток байтов называется байтовой *сериализацией*. Обратный процесс считывания из линейного потока и формирования объекта называется *десериализацией*.

Часто сохранение данных с использованием сериализации выливается в код меньшего объема.

Сериализация

Объект может содержать ссылки на другие объекты, которые в свою очередь содержат ссылки на другие объекты (и т.д.).

Для описания этих сложных связей необходимо построение так называемого `графа объектов` (или `дерева объектов`).

При сериализации сохраняется сам граф объектов и состояние всех объектов, входящих в граф.

При десериализации по считанному графу объектов восстанавливаются их связи и состояния.

Построение графа объектов и настройка сериализации производится автоматически.

...

Для того, чтобы класс был сериализуем, нужно указать для него атрибут `[Serializable]`.

Классы, помеченные этим атрибутом, могут быть сериализованы, то есть представлены в виде потока байтов.

Сериализация применяется к свойствам и полям класса. Если мы не хотим, чтобы какое-то поле класса сериализовалось, то мы его помечаем атрибутом `[NonSerialized]`.

Пример

```
[Serializable]
class Person
{
    public string Name { get; set; }
    public int Year { get; set; }

    [NonSerialized]
    public string accNumber;

    public Person(string name, int year, string acc)
    {
        Name = name;
        Year = year;
        accNumber = acc;
    }
}
```

Пример

При наследовании подобного класса, следует учитывать, что атрибут `Serializable` автоматически не наследуется.

И если необходимо, чтобы производный класс также мог бы быть сериализован, то опять же мы применяем к нему атрибут:

```
[Serializable]
class Worker : Person
```

Двоичные потоки данных

Процессом сериализации занимается специальный внешний объект класса `Formatter`.

Существует два типа сериализации:

- байтовая,
- XML формат.

Рассмотрим только байтовую сериализацию.

Двоичный форматер описан в пространстве имен `System.Runtime.Serialization.Formatters.Binary` и принадлежит классу `BinaryFormatter`.

Класс BinaryFormatter

Тип `BinaryFormatter` сериализует состояние объекта в поток, используя компактный двоичный формат.

Этот тип определен в пространстве имен `System.Runtime.Serialization.Formatters.Binary`.

Таким образом, чтобы получить доступ к этому типу, необходимо указать следующую директиву `using`:

```
// Получить доступ к BinaryFormatter:
using System.Runtime.Serialization.
Formatters.Binary;
```

Класс BinaryFormatter. Основные методы

`Serialize()` – сериализация объекта в байтовый поток.

`Deserialize()` – десериализация объекта из байтового потока.

Пример

```
[Serializable]
class Class1
{    int n;      …   }


[Serializable]
class Class2
{    Class1 c1 = new Class1();…    }
…


Class2 c2 = new Class2();
```

Пример

```
FileStream fs = File.Create(”my.bin”);

BinaryFormatter bf = new BinaryFormatter();
bf.Serialize(fs, c2);
fs.Close();

fs = File.OpenRead(”my.bin”);
Class2 c3 = (Class2)bf.Deserialize(fs);
fs.Close();
…
```

Пример. Комментарий

Форматтер десериализует объект из потока и возвращает на него ссылку типа `Object`, поэтому необходимо явное преобразование к реальному типу объекта.

Если какие-то поля класса не нужно сохранять при сериализации, то их помечают атрибутом `NonSerialized`.

При десериализации в несериализуемые поля объекта записываются значения по умолчанию.

Не сериализуются также константы и статические поля, поскольку они не принадлежат объекту.

Ссылки, источники…

1. Formatter Class. (Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/api/system.runtime.serialization.formatter?view=netcore-2.1).
2. BinaryFormatter Class (Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/api/system.runtime.serialization.formatters.binary.binaryformatter?view=netcore-2.1)
3. Сериализация. (Режим доступа: https://professorweb.ru/my/csharp/thread_and_files/level4/4_1.php).