Ввод-вывод в C#

Сериализация

Сохранение объектов в .NET

В терминах .NET **сериализация** (serialization) — это термин, описывающий процесс преобразования объекта в линейную последовательность байтов. Обратный процесс, когда из потока байтов, содержащего всю необходимую информацию, объект восстанавливается в исходном виде, называется **десериализацией** (deserialization).

Сохранение объектов в .NET

Вот некоторые основные области, где сериализация дает преимущества:

- *Доступность* - компонент можно сохранить в файле и обращаться к нему в любое время.
- *Время жизни* - сохранение объекта с его состоянием продлевает ему жизнь. Обычно, когда вы закрываете приложение, все связанные с ним объекты автоматически уничтожаются.
- *Использование в сетевых приложениях* - сложная форма объекта была преобразована в формат, подходящий для передачи через сеть.
- *Надежность* – сохраненный объект можно воссоздать "как он есть".

Сериализация в формат XML

Сериализация объекта в формат XML имеет определенные преимущества - в первую очередь вы трансформируете специфичную для системы информацию о состоянии в текст, который можно легко переслать по сети и через брандмауэры.

Однако в полученных XML данных не сохраняются типы разнообразных используемых полей, вместо этого свойства, поля и возвращаемые значения сериализуются в XML формат. Эта особенность полезна, если нужно передавать значения, а не детальную информацию о самом объекте.

Класс XmlSerializer из пространства имен System. Xml. Serialization обеспечивает функциональные возможности сериализации и десериализации объектов в XML формате.

Сериализация в формат XML

Для сериализации класса есть два простых правила:

- Класс должен поддерживать используемый по умолчанию открытый конструктор без параметров. Это требование связано с тем, что при воссоздании объекта через процесс десериализации сначала экземпляр объекта создается конструктором по умолчанию, а затем из входящего потока данных устанавливаются открытые свойства. Если конструктор по умолчанию отсутствует, .NET Framework не будет знать, как создать объект.

- Сохраняются только открытые свойства, поддерживающие операции get и set, и открытые члены данных. Это объясняется тем, что процесс сериализации не может обращаться к закрытым и доступным только на чтение элементам данных.

Сериализация в формат XML

Для сериализации следует подключить пространство имен
System.Xml.Serialization

Предположим, у нас есть класс Customer.

```
Customer cust1 = new Customer();
…
// создаем поток для записи
    StreamWriter writer = new StreamWriter("Customer.xml");
// создаем сериализатор
    XmlSerializer serializer = new XmlSerializer(typeof(Customer));
// и сериализуем объект
    serializer.Serialize(writer, cust1);
    writer.Close();
. . .
```

Десериализация из формата XML

```
. . .
// создаем поток для записи
    FileStream reader= new FileStream("Customer.xml",
    FileMode.Open, FileAccess.Read);
// создаем десериализатор
      XmlSerializer deserializer = new XmlSerializer(typeof(Customer));
// и десериализуем объект
          Customer cust2 = (Customer)deserializer.Deserialize(reader);.
. . .
```

Сериализация с помощью объектов форматирования

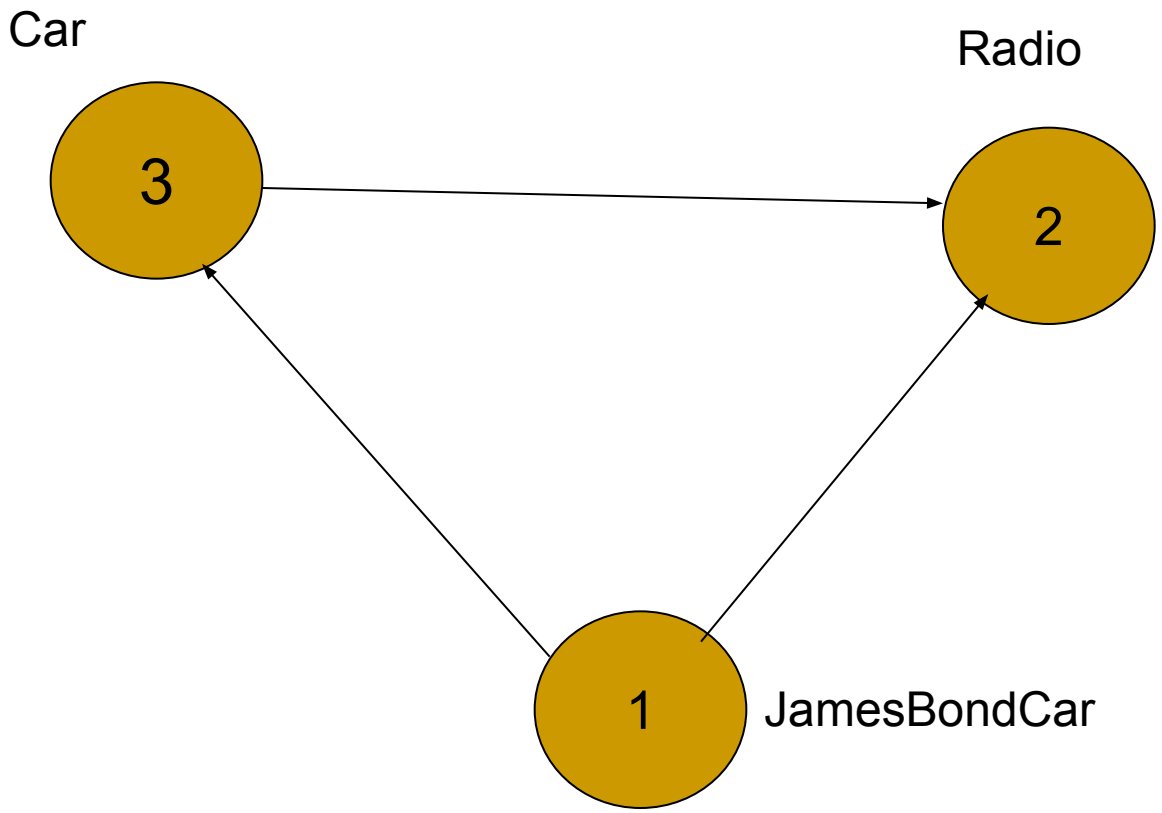
Чтобы можно было провести сериализацию объекта, каждый класс, который будет участвовать в сериализации, должен обладать атрибутом **[Serializable].**

```
// Класс Customer может быть сериализован
[Serializable]
public class Customer
{
    public int ID;
       public String FIO;
       public DateTime Date;
       public decimal Credit;
    // Однако нам нет необходимости сохранять это число
       [NonSerialized]
       public decimal CurrentPayment;
}
```

Сохранение объектов в .NET

Службы сериализации в .NET — это весьма сложные программные модули, Они обеспечивают многие неочевидные вещи: например, когда объект сериализуется в поток, информация о всех других объектах, на которые он ссылается, также должна сериализоваться. Например, когда сериализуется производный класс, ссылки на другие классы, которые есть в базовых классах для этого производного класса, также должны отслеживаться и учитываться.

Графы для отношений объектов



[Car 3, ref 2], [Radio 2], [JamesBondCar 1, ref 3, ref 2]

Графы для отношений объектов

Набор взаимосвязанных объектов, сериализованных в поток, называется графом объектов (object graph). Графы позволяют фиксировать отношения объектов друг к другу, и они *не соответствуют* классическим моделям отношений классов в объектно-ориентированном программировании.

Внутри графа каждому из объектов присваивается уникальный номер, который используется только для служебных целей самого графа и которому совершенно не обязательно должно что-то соответствовать в реальном мире. Далее записывается информация о соответствии имени класса этому номеру, информация о всех отношениях этого класса с другими классами и отношениях других классов между собой.

Выбираем объект Formatter

После того как мы пометили класс как доступный для сериализации, наша следующая задача — выбрать формат, в котором будет сохранен объектный граф.

Пространство имен System.Runtime.Serialization.Formatters включает в себя еще два пространства имен - *.Binary и *.Soap, каждому из которых соответствует один из двух объектов Formatter, которые можно использовать по умолчанию.

Класс BinaryFormatter сериализует объектный граф в компактном потоке двоичного формата,

в то время как класс SoapFormatter представляет граф как сообщение протокола SOAP (Simple Object Access Protocol — простого протокола доступа к объектам) в формате XML.

Выбираем объект Formatter

Класс BinaryFormatter определен в библиотеке mscorlib.dll, поэтому единственное, что нам потребуется для сериализации при помощи объекта Formatter, — определить использование этого пространства имен:

// Для сериализации объектов в двоичном формате

```
using System.Runtime.Serialization.Formatters.Binary;
```

Класс SoapFormatter определен в отдельной сборке, поэтому для сохранения объекта в формате SOAP вам вначале потребуется добавить ссылку на сборку System.Runtime.Serializaton.Formatters.Soap.dll,а затем использовать аналогичную команду:

// Для сериализации объектов в формате SOAP

```
using System.Runtime.Serialization.Formatters.Soap;
```

Сериализация в двоичном формате

```
public static void Main()
{
// Создаем объект JamesBondCar и выполняем с ним всякие
// действия
JamesBondCar myAuto = new JamesBondCar(“Fred“, 50, false, true);
myAuto.TurnOnRadio(true);
myAuto.GoUnderWater();

// Создаем поток для записи в файл
FileStream myStream = File.Create("CarData.dat");

// Помещаем объектный граф в поток в двоичном формате
BinaryFormatter myBinaryFormat = new BInaryFormatter();
myBinaryFormat.Serialize(myStream, myAuto);
myStream.Close();
. . .
}
```

Сериализация в формате SOAP

```
using System.Runtime.Serialization.Formatters.Soap;
. . .
// Сохраним тот же самый объект в формате SOAP
FileStream myStream = File.Create(“CarData.xml");
SoapFormatter myXMLFormat = new SoapFormatter();
myXMLFormat.Serialize(myStream,  myAuto);
myStream.Close();

// Восстанавливаем объект из файла SOAP
myStream = File.OpenRead(“CarData.xml");
JamesBondCar carFromXML =
    (JamesBondCar)myXMLFormat.Deserialize(myStream);
Console.WriteLine(carFromXML.PetName + " is alive!");
myStream.Close();
. . .
```

Задание

Напишите программу, которая сериализует и десериализует произвольный объект в файл. Используйте обычный формат XML и формат SOAP. Просмотрите полученные XML-файлы в любом текстовом редакторе или браузере.