对象序列化

北京理工大学计算机学院
金旭亮

对象的“状态”

- 对象序列化主要解决的是**对象状态**的保存问题。
- “**对象状态**”，可以看成是某一时刻对象所拥有的各个字段值的集合。
- 对象状态是与时间关联在一起的，在不同的时刻，由于字段值的变化，对象可能会处于不同的状态。

对象的序列化

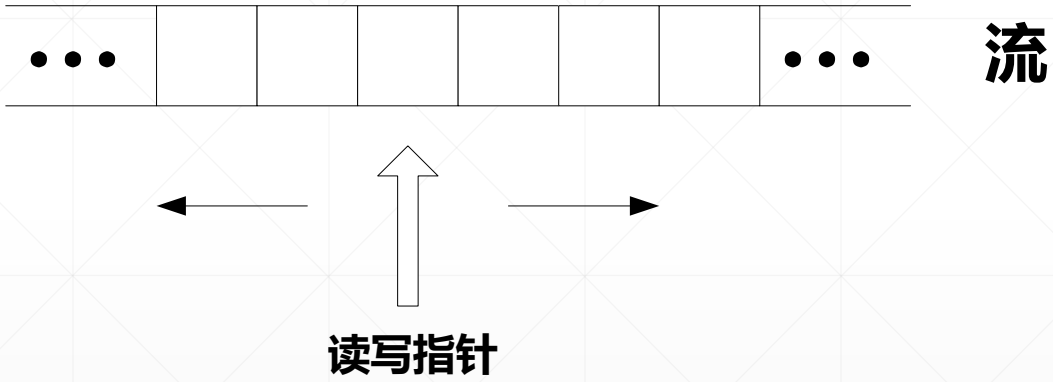
- 将对象状态保存到另外一种媒介中，并在需要时可以从媒介中重新读取数据重建对象的过程称为对象的“**序列化（Serialization）**”与“**反序列化（Deserialization）**”。

在开发中，用于保存对象状态的常用媒介有：

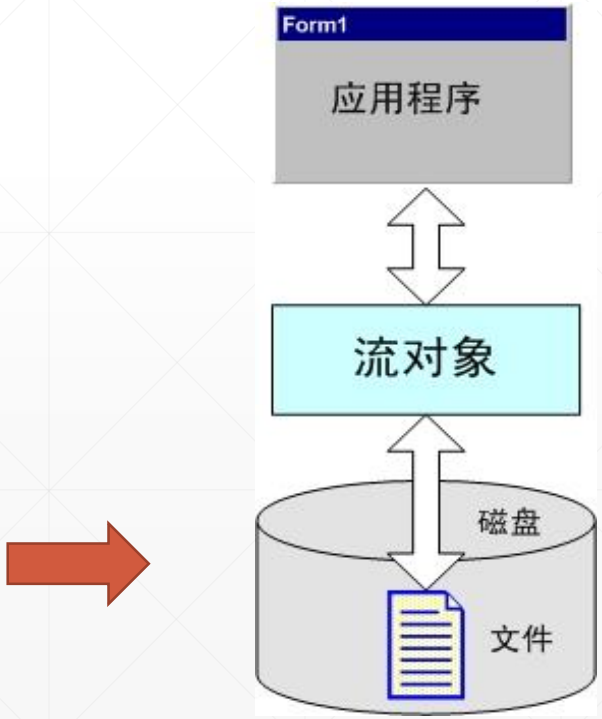
1. “流（Stream）”
2. “字符串（String）”

什么是“流”？

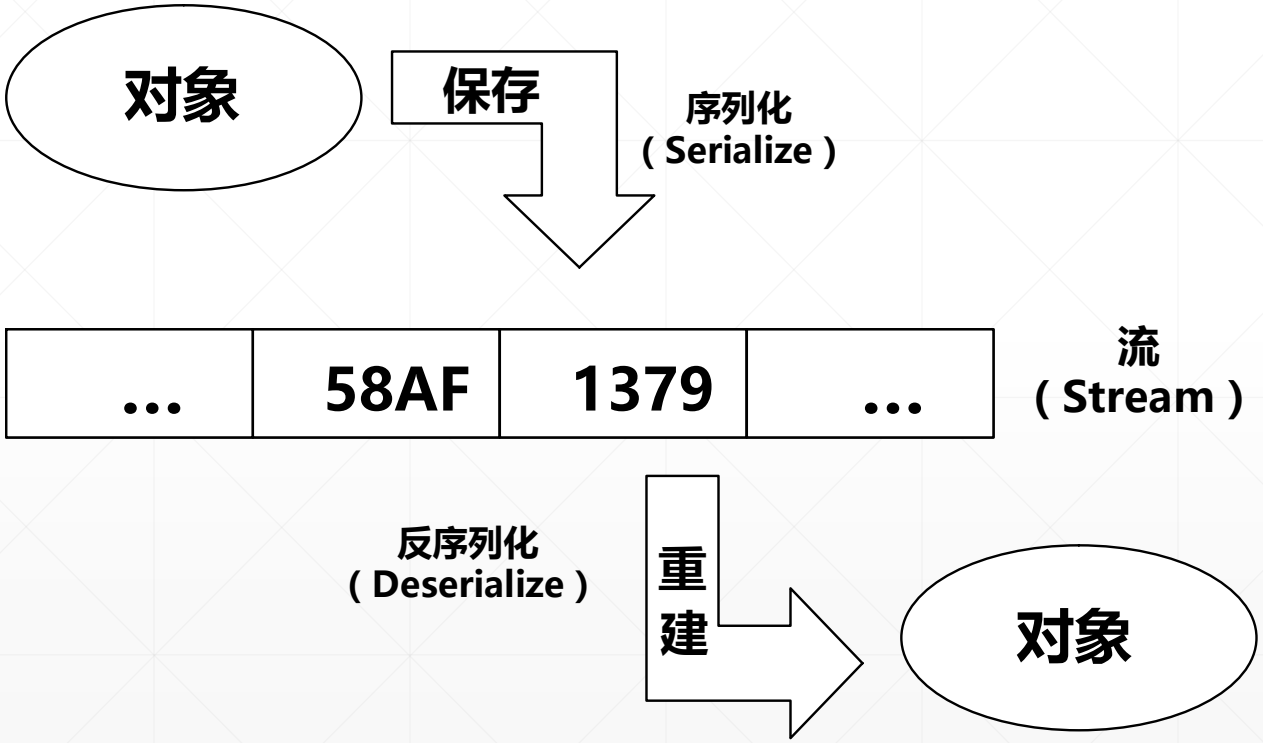
“流（Stream）”是一个抽象的概念，它代表的是一连串有顺序的二进制数据。



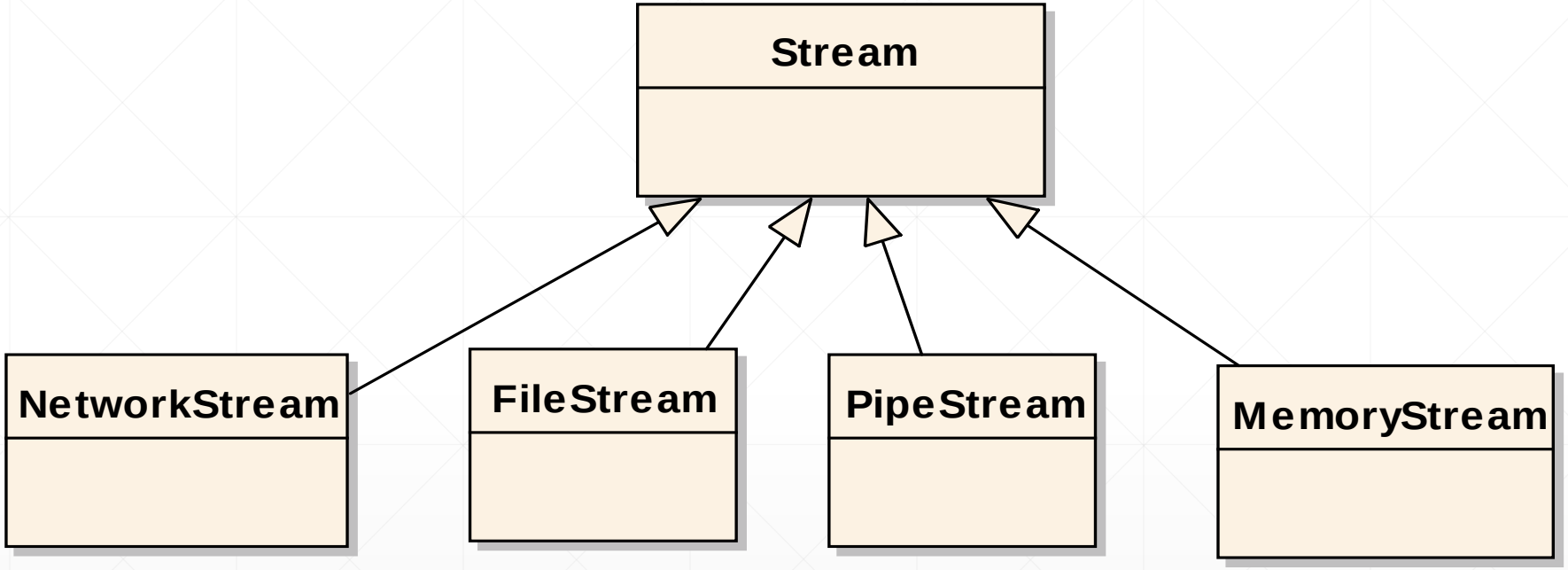
有多种类型的流，比如一个打开的文件就可以看成是一个流，称为“文件流（File Stream）”。



使用流实现序列化


对象
保存
序列化
(Serialize)
...
58AF
1379
...
流
(Stream)
反序列化
(Deserialize)
重建
对象

.NET基类库提供的几种流对象



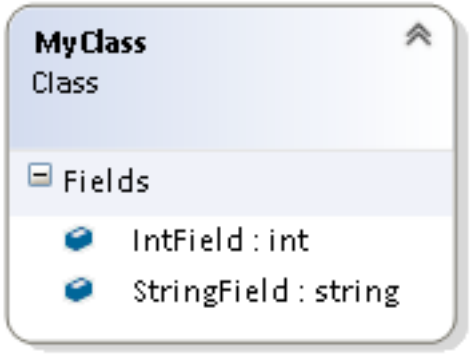
两种对象序列化方式

- **二进制序列化**：将对象的数据看成是二进制的数据而直接写入到流中。
- **XML序列化**：另一种是将对象数据用XML方式表示之后再以纯文本的方式写入到流中。

完成序列化工作的是“格式化器（Formatter）”：

1. BinaryFormatter完成二进制序列化工作
2. SoapFormatter完成XML序列化工作

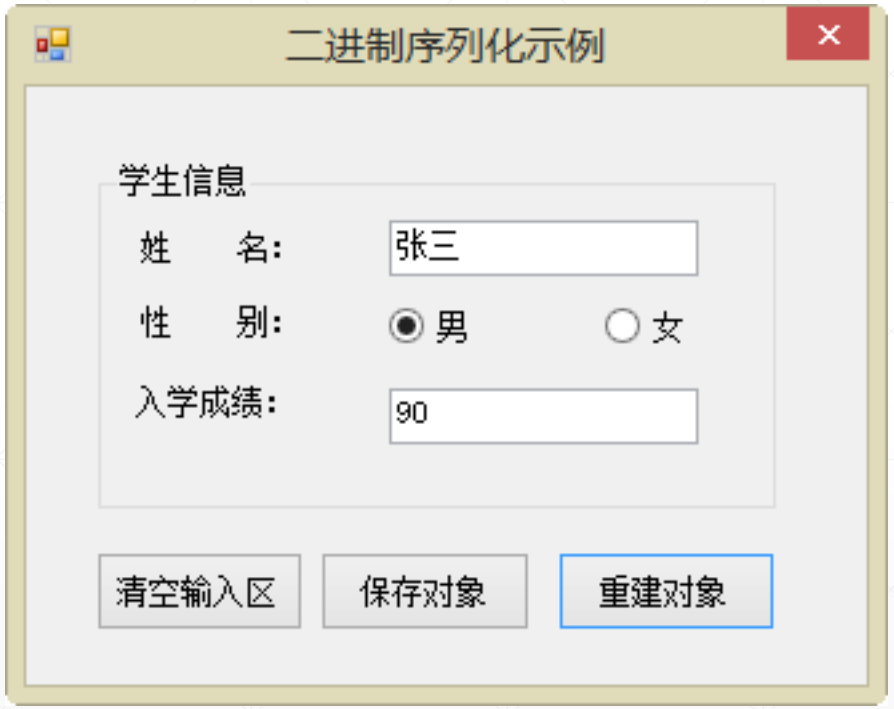
XML格式化实例




```
static void Main(string[] args)
{
    var Serializer = new XmlSerializer(typeof(MyClass));
    MyClass obj = new MyClass() {
        IntField = 100,
        StringField = "Hello" };
    Serializer.Serialize(Console.Out, obj);
    Console.ReadKey();
}
```

示例：UseXmlSerializer

二进制序列化实例



支持二进制序列化的类要求拥有[Serializable]标记

```
//学生信息
[Serializable]
7 references
class CollegeStudent
{
    //姓名
    public String Name = "空";
    //性别
    public bool IsMale = true;
    //入学考试成绩
    public int ScoreForEntranceExamination = 0;
}
```

```
//将CollegeStudent对象序列化到文件中
1 reference
private void SerializeObj(String FileName,CollegeStudent stu)
{
    //创建FileStream对象
    using (FileStream writer = new FileStream(FileName, FileMode.Create))
    {
        //创建格式化器对象
        IFormatter formatter = new BinaryFormatter();
        //格式化器对象使用FileStream对象序列化对象
        formatter.Serialize(writer, stu);
        MessageBox.Show("对象成功保存到文件:" + FileName);
    }
}
```

二进制序列化采用BinaryFormatter对象的Serialize()方法完成

反序列化

```
//从文件中反序列化对象
1 reference
private CollegeStudent  DeserializeObj(String FileName)
{
    using (FileStream reader = new FileStream(FileName, FileMode.Open))
    {

        IFormatter formatter = new BinaryFormatter();
        return (CollegeStudent)formatter.Deserialize(reader);
    }
}
```

二进制反序列化采用BinaryFormatter对象的DeSerialize()方法完成，此方法返回一个Object类型的对象，通常还需要进行类型转换。

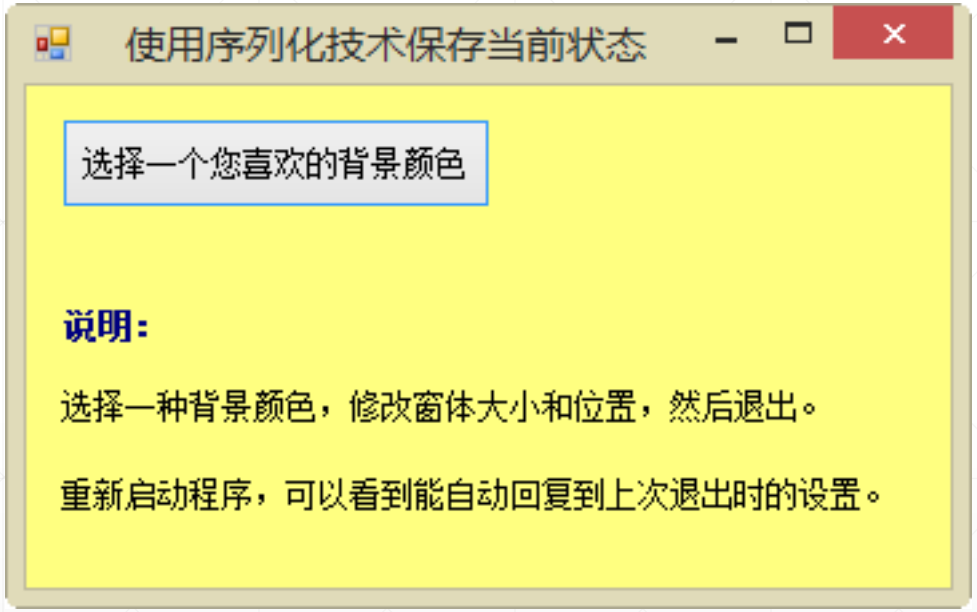
应用实例：大批地复制对象

```
MyClass obj = new MyClass();
//创建一个内存流对象
using (MemoryStream ms = new MemoryStream())
{
    IFormatter formator = new BinaryFormatter();
    //将对象序列化到内存流中
    formator.Serialize(ms, obj);
    //克隆100个对象
    for (int i = 0; i < 100; i++)
    {
        //回到流的开头
        ms.Seek(0, SeekOrigin.Begin);
        //反序列化对象
        obj = (formator.Deserialize(ms) as MyClass);
        obj.Index += i;   //设置对象字段
        Console.WriteLine("对象{0}已创建。", obj.Index);
    }
}
```

先将对象序列化到内存流，然后将流的读写指针移回开头，再反序列化，即可创建一个与原有对象“一模一样”的对象。
重复这个过程，可以克隆多个对象。

示例:ObjectCloneViaSerialization

应用实例：程序退出时保存状态



SaveFormStatus示例

- 设计一个FormStatus类封装窗体背景和大小位置信息。
- 程序退出时将FormStatus对象序列化到FormStatus.cfg配置文件中。
- 程序重启时反序列化它即可。