序列化是将对象或对象图转换成字节流的过程，反序列化是将字节流转换回对象图的过程。

在序列化和反序列化的上下文中，对象图是指一组对象及其之间的关系。当你序列化一个对象时，不仅该对象本身会被序列化，而且所有由该对象引用的其他对象也会被递归地序列化。这个由所有这些对象构成的网络就是所谓的“对象图”。

二进制序列化：

文本

低可信度描述已自动生成

电脑屏幕的照片上有文字

描述已自动生成

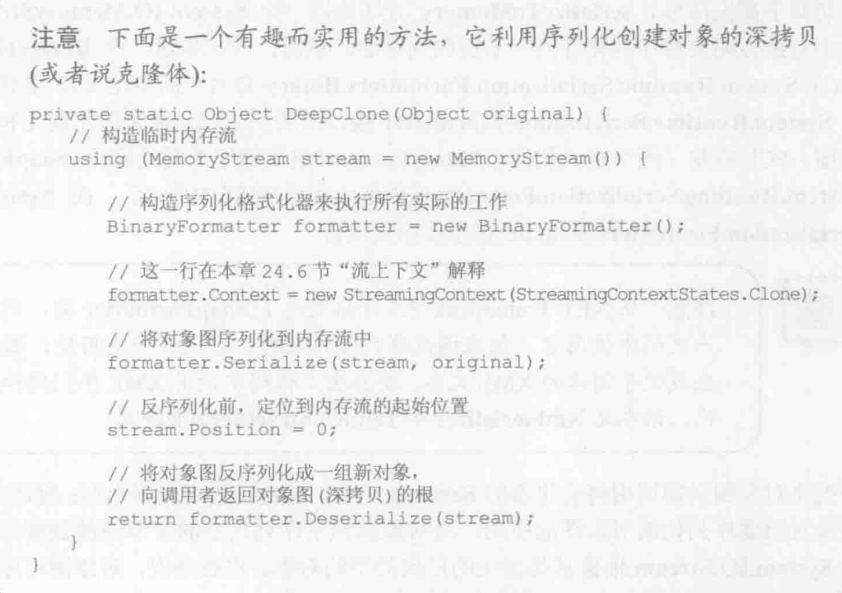
BinaryFormatterer类：

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

[c#XML、JSON的序列化和反序列化，看完你就懂了\_jsonconvert 序列化xml\_naer\_chongya的博客-CSDN博客](https://blog.csdn.net/naer_chongya/article/details/130350226)

利用序列化完成深拷贝：



使用序列化和反序列化注意事项和操作：

1.序列化和反序列化使用相同的序列化器

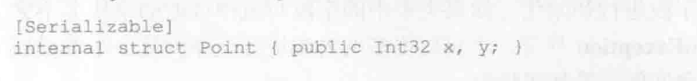
2.可以将多个对象图序列化到一个流中

文本

描述已自动生成

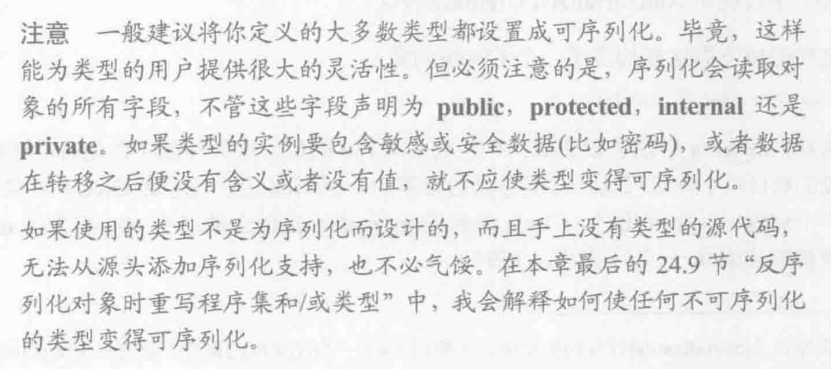
3.序列化对象时，类型的全名和类型定义的程序集的全名会被写入流。

使得类型的对象可序列化需要在类型的声明前加上Serializable特性



Serializable定制特性只能用于class、struct、enum和delegate，且其中enum和delegate类型默认是可序列化的不用显式添加Serializable特性

Ps：不存在基类不可序列化但是子类的可以序列化的情况。（大概可以通过对象图的概念来理解？？）



可序列化的类型中的字段，如果使用了NonSerialized特性在，那么这个字段那就不会被序列化，且这个特性会被继承.

但是为了使得不可被序列化的字段在反序列化后仍然是正确的值，可以使用其他可序列化的字段为它赋值，需要使用OnDeserialized方法，这是OnDeserialized特性提供的方法，可以在完成反序列化之后赋值

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

这个特性还提供了另外三个方法，在序列化前和反序列化后进行一些操作：

文本

描述已自动生成

类型对象序列化和反序列化顺序

手机屏幕截图

描述已自动生成

报纸上的文字

描述已自动生成

格式化器如何序列化类型实例：

步骤：

1.格式化器调用FormatterServices的GetSerializableMembers方法

文本

描述已自动生成

2.对象被序列化，MemberInfo对象数组传给FormatterServices的静态方法GetObjectData

文本

描述已自动生成

3.格式化器将程序集标识和类型的完整名称写入流中

4.格式化器然后遍历两个数组中的元素，将每个成员的名称和值写入流中

格式化器如何反序列化类型实例：

1.格式化器从流中读取程序集标识和完整类型名称。

一些文字和图片的手机截图

描述已自动生成

2.格式化器调用FormatterServices的静态方法GetUninitializedObject

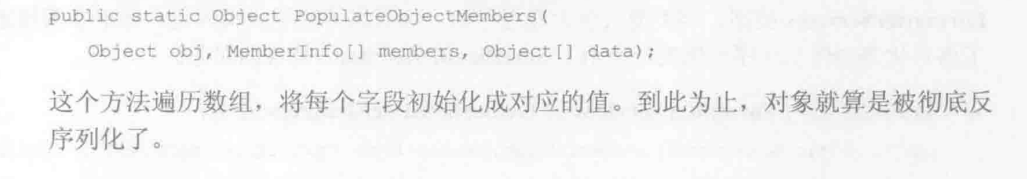
文本

描述已自动生成

3.格式化器构造并初始化一个MemberInfo数组

4.格式化器根据数据流中包含的数据创建并初始化一个Object数组

5.将新分配对象、MemberInfo数组以及并行Object数组的引用传给FormatterServices的PopulateObjectMembers静态方法



控制序列化/反序列化的数据

如果使用格式化器，格式化器使用反射，因此耗时多性能差

文本

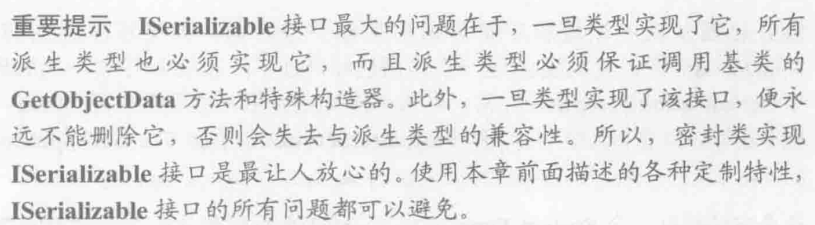
描述已自动生成

继承这个接口就可以避免反射

图片包含 信件

描述已自动生成

Ps：



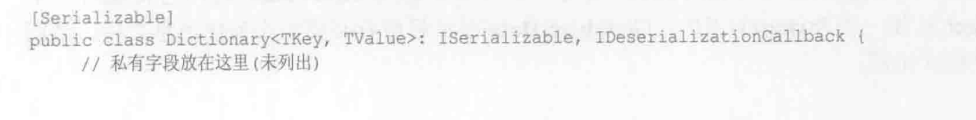
所有派生类必须实现这个接口，可以用对象图的概念去理解，如果一个类实现了ISerializable接口，那么它就有责任序列化自己的所有字段，包括从基类继承的字段。如果派生类没有实现ISerializable接口，那么基类的字段就不会被正确序列化

GetObjectData方法：它用于在序列化过程中填充SerializationInfo对象，以便将对象序列化所需的数据存储到流中。当一个对象被序列化时，格式化器会自动调用该对象的GetObjectData方法，以获取序列化所需的数据。

SerializationInfo类：

SerializationInfo类用于存储序列化或反序列化对象所需的所有数据。这个类通常由具有自定义序列化行为的对象使用。在序列化过程中，ISerializable.GetObjectData或ISerializationSurrogate方法使用每条要序列化的信息的名称、类型和值来填充SerializationInfo存储区。在反序列化过程中，相应的函数可以提取此信息。对象在序列化时使用AddValue方法将数据添加到SerializationInfo存储区，并在反序列化时使用GetValue方法从SerializationInfo存储中提取对象。这个类提供了一系列的AddValue和GetValue方法，用于添加和获取各种类型的数据，包括布尔值、整数、浮点数、字符串等。此外，这个类还提供了一些属性，如AssemblyName、FullTypeName、MemberCount和ObjectType等，用于获取序列化过程中的一些信息。

例子：

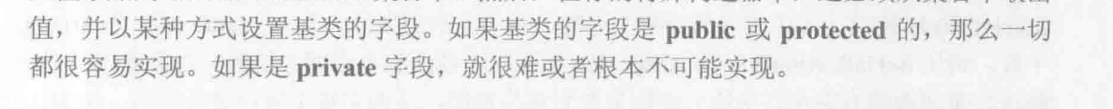


电脑屏幕的照片上有文字

描述已自动生成

如果基类没有实现ISerializable，派生类需要手动序列化基类字段并添加到SerializationInfo中

Ps：



例子：

电脑屏幕的照片上有文字

描述已自动生成

文本

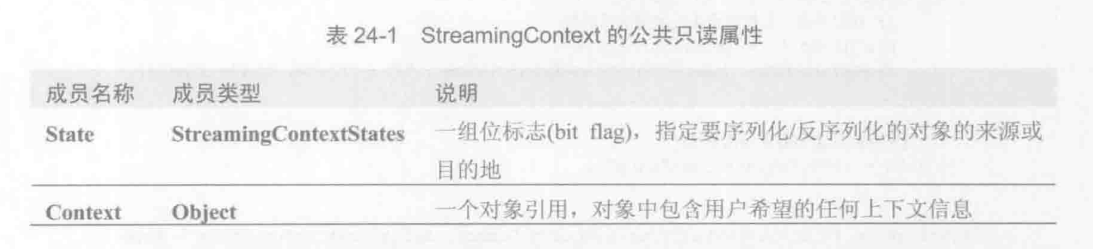
描述已自动生成

文本

描述已自动生成

流上下文：

StreamingContext的结构：



表格

描述已自动生成

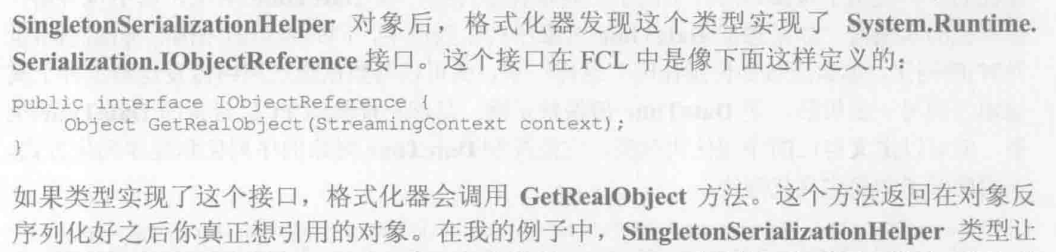
使用方法：

报纸上的文字

描述已自动生成

类型序列化为不同类型或者对象反序列化为不同类型的对象：

P560-561代码讲解很详细



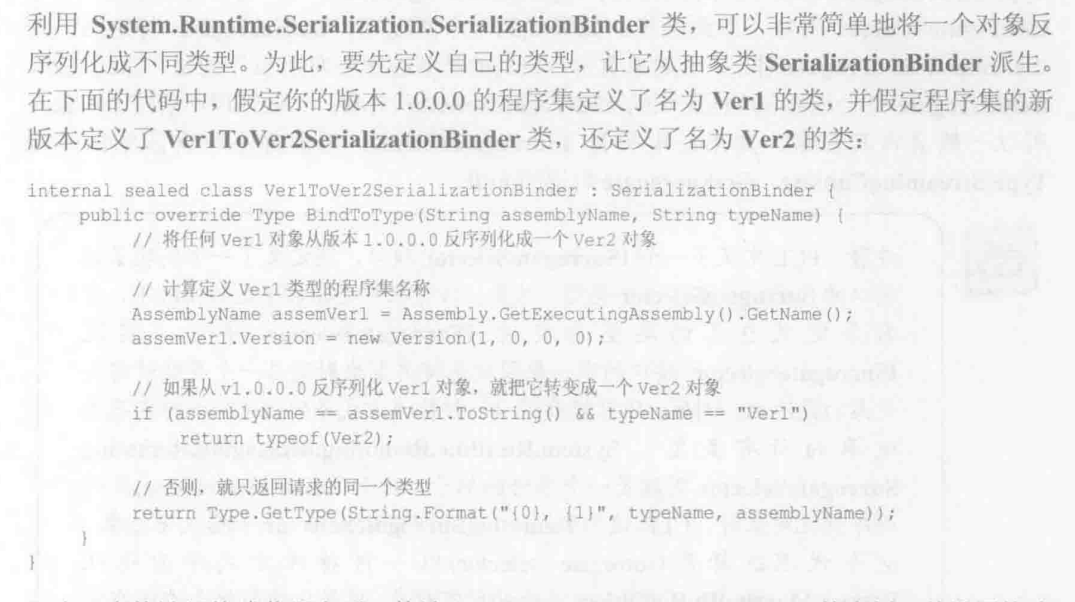
实现这个接口之后，在反序列化后会自动调用

图片包含 文本

描述已自动生成

反序列化对象时重写程序集/类型（没看懂，没理解）

把对象反序列化成序列化时不同的类型



重写的BindToType在反序列化时自动调用，BindToName也是