维基百科
自由的百科全书

序列化

序列化（serialization）在计算机科学的资料处理中，是指将数据结构或对象状态转换成可取用格式（例如存成文件，存于缓冲，或经由网络中发送），以留待后续在相同或另一台计算机环境中，能恢复原先状态的过程。依照序列化格式重新获取字节的结果时，可以利用它来产生与原始对象相同语义的副本。对于许多对象，像是使用大量引用的复杂对象，这种序列化重建的过程并不容易。面向对象中的对象序列化，并不概括之前原始对象所关系的函数。这种过程也称为对象编组（marshalling）。从一系列字节提取数据结构的反向操作，是**反序列化**（也称为解编组、deserialization、unmarshalling）。

序列化在计算机科学中通常有以下定义：

- 对同步控制而言，表示强制在同一时间内进行单一访问。
- 在数据储存与发送的部分是指将一个对象存储至一个存储介质，例如文件或是存储器缓冲等，或者透过网络发送资料时进行编码的过程，可以是字节或是XML等格式。而字节的或XML编码格式可以还原完全相等的对象。这程序被应用在不同应用程序之间发送对象，以及服务器将对象存储到文件或数据库。相反的过程又称为反序列化。

用途

- 经由电信线路传输资料的方法（通信）。
- 存储资料的方法（在数据库或硬盘）。
- 远程程序调用的方法，例如在SOAP中。
- 在以组件为基础，例如COM，CORBA的软件工程中，是对象的分布式方法。
- 检测随时间资料变动的方法。

为了达成上述功能其一能有效作用，则必须与硬件结构保持独立性。譬如说为了能最大化分布式的使用，在不同硬件运行的计算机，应该能够可靠地重建序列化资料流，而不依赖于字节序。虽然直接复拷存储器中的数据结构更简便又快速，可是对于其它不同硬件的机器，却无法可靠地运作。以独立于硬件之外的格式来序列化数据结构，要避开字节序、存储器布局、或在不同编程语言中数据结构如何表示等等之类的问题。

对于任何序列化方案的本质来说，因为资料编码是根据定义连续串在一起的，提取序列化数据结构中的某一部分，则需要从头到尾读取整个对象并且重新建构。这样的资料线性在许多应用中是有利的，因为它使输出入接口简单而共同，能被用来保持及传递对象的状态。

要求高性能的应用时，花费精力处理更复杂的非线性存储系统是有其必要意义的。即使在单一机器上，原始的指针对象也非常脆弱无法保存，因为它们指向的标地可能重新加载到内存中的不同地址。为了处理这个问题，序列化过程包括一个步骤：将引用的直接指针转换为以名称或位置的间接引用，称之为不挥发（unswizzling）或者指针不挥发。反序列化过程则包括了称为指针旋转（swizzling）的反向步骤。由于序列化和反序列化可从共通代码（例如，微软MFC中的*Serialize*函数）驱动，所以共通代码可同时进行两次，因此，

1. 检测要序列化的对象与其先前副本之间的差异，

2. 提供下一次这种检测的输入。因为差异可以被即时检测，所以不必再重新建立先前的副本。该技术称为差异分辨执行。

这技术应用在内容随时间变化的用户界面编程中－依照输入事件来处理图形对象的产生、移除、更改或制作，而无需编写另外的代码执行这些操作。

缺点

序列化可能会破解抽象资料类型的封装实现，而使其详细内容曝光。简单的序列化实现可能违反面向对象中私有资料成员需要封装（encapsulation）的原则。商用软件的出版商通常会将应用软件的序列化格式，当作商业秘密，以阻碍竞争对手生产可兼容的产品；有些会蓄意地混淆，或甚至将序列化资料作加密处理。然而，互通可用性的要求应用程序能够理解彼此的序列化格式。因此，像CORBA的远程方法调用架构详细定义了它们的序列化格式。许多机构，例如档案馆和图书馆，尝试将他们的备份文件－特别是数据库抛档（dump），存储成一些相对具可读性的序列化格式中，使备份资料不因信息技术变迁而过时。

序列化格式

20世纪80年代初的施乐网络系统快递技术影响了第一个广泛采用的标准。Sun Microsystems在1987年发布了外部数据表示法（XDR）。90年代后期开始推动标准序列化的协议：XML（可扩展标记语言）应用于产生人类可读的文字编码。资料以这样的编码使存续的对象能有效用，无论相对于人是否可阅读与理解，或与编程语言无关地传递给其它信息系统。它缺点是失去了扎实的编码字节流，但截至目前技术上所提供大量的存储和传输容量，使得文件大小的考量，已不同于早期计算机科学的重视程度。

二进制XML被提议作为一种妥协方式，它不能被纯文本编辑器读取，但比一般XML更为扎实。在二十一世纪的Ajax技术网页中，XML经常应用于结构化资料在客端和服务端之间的异步传输。相较于XML，JSON是一种轻量级的纯文字替代，也常用于网页应用中的客端－服务端通信。JSON肇基于JavaScript语法所派生，但也广为其它编程语言所支持。与JSON类似的另一个替代方案是YAML，它包含加强序列化的功能，更“人性化”而且更扎实。这些功能包括标记资料类型，支持非层次结构式数据结构，缩进结构化资料的选项以及多种形式的标量资料引用的概念。

另一种可读的序列化格式是属性列表（property list）。应用在NeXTSTEP、GNUstep和macOS Cocoa环境中。

针对于科学使用的大量资料集合，例如气候，海洋模型和卫星数据，已经开发了特定的二进制序列化标准，例如HDF，netCDF和较旧的GRIB。

编程语言支持

一些面向对象的编程语言直接支持对象序列化（或对象归档），可借由语法糖元素或者提供了标准接口。这些编程语言其中有Ruby，Smalltalk，Python，PHP，Objective-C，Delphi，Java 和.NET系列语言。若是缺少原生支持序列化的编程语言，也可使用额外的函数库来添加功能。

C/C++

C 和 C++ 没有提供任何类型的高阶序列化构造，但是两种语言都支持将内置资料类型以及一般的数据结构(struct)输出为二进制资料。因此，开发人员自己定义序列化函数是显而易举的。此外，基于编译器的解决方案，如用于 C++ 的ODB ORM系统，能够自动产生类声明的序列化源码，不必修改或仅少量的修改。其它普及的序列化框架是有来自Boost框架的Boost.Serialization，S11n和Cereal等框架。微软的MFC框架也提供序列化方法，作为其文件视图(Document-View)架构的部件。

Java

Java 提供自动序列化，需要以java.io.Serializable接口的实例来标明对象。实现接口将类别标明为“可序列化”，然后Java在内部处理序列化。在Serializable接口上并没有预先定义序列化的方法，但可序列化类别可任意定义某些特定名称和签署的方法，如果这些方法有定义了，可被调用执行序列化/反序列化部分过程。该语言允许开发人员以另一个Externalizable接口，更彻底地实现并覆盖序列化过程，这个接口包括了保存和恢复对象状态的两种特殊方法。

在默认情况下有三个主要原因使对象无法被序列化。其一，在序列化状态下并不是所有的对象都能获取到有用的语义。例如，Thread对象绑定到当前Java虚拟机的状态，对Thread对象状态的反序列化环境来说，没有意义。其二，对象的序列化状态构成其类别兼容性缔结（compatibility contract）的某一部分。在维护可序列化类别之间的兼容性时，需要额外的精力和考量。所以，使类别可序列化需要慎重的设计决策而非默认情况。其三，序列化允许访问类别的永久私有成员，包含敏感信息（例如，密码）的类别不应该是可序列化的，也不能外部化。上述三种情形，必须实现Serializable接口来访问Java内部的序列化机制。标准的编码方法将字段简单转换为字节流。

原生类型以及永久和非静态的对象引用，会被编码到字节流之中。序列化对象引用的每个对象，若其中未标明为transient的字段，也必须被序列化；如果整个过程中，引用到的任何永久对象不能序列化，则这个过程会失败。开发人员可将对象标记为暂时的，或针对对象重新定义的序列化，来影响序列化的处理过程，以截断引用图的某些部分而不序列化。Java并不使用构造函数来序列化对象。

由JDBC也可对Java对象进行序列化，并将其存储到数据库中。虽然Swing组件的确实例化了Serializable接口，但它们不能移植到有版本差异的Java虚拟机之间。因此，Swing组件或任何继承它的组件可以序列化为字节数组，但不能保证这个仓存在另一台机器上可读取。

Perl

由CPAN所提供的几个Perl模块提供序列化机制，包括了Storable，JSON::XS和FreezeThaw。Storable包括将文件或Perl标量的数据结构，将其序列化和反序列化的功能。除了直接序列化到文件之外，Storable 还包含了冻结功能，将包装为标量的资料，返回其序列化的副本；并可用thaw(解冻)这个标量来反序列化。这对于以网络插座（socket）发送复杂的数据结构，或将其存储于数据库中非常有用。

当利用Storable对结构进行序列化时，具备了网络安全性的功能，它们以降低一点性能的成本，将资料存储为任何计算机可读取的格式。这些功能的名称有nstore，nfreeze等。依硬件特定的，带字母“n”函数所序列化的这些结构，则以没有字母“n”的函数将之反序列化－常态地解冻并截取反序列化结构。

PHP

PHP最初通过内置的`serialize()`和`unserialize()`函数来实现序列化。PHP可以序列化任何其它数据类型，除了资源（文件指针，socket等）以外。对不受信任的资料上使用内置的`unserialize()`函数时，通常是有风险的。对于对象有两种“魔术方法”，`__sleep()`和`__wakeup()`，可以在类别中实现。而会分别从`serialize()`和`unserialize()`中调用，对应于清理和恢复对象的功能。例如，在序列化时可能需要关闭数据库连线，并在反序列化时恢复连线；这个功能可在这两种魔术方法中处理。它们也允许对象选择哪些属性可被序列化。从PHP 5.1开始有面向对象的序列化机制，即为`Serializable`接口。

Python

Python编程核心的序列化机制是`pickle`标准函数库，这名称暗示数据库相关的特别术语“浸渍”，来描述资料反序列化（unpickling for deserializing）。Pickle 使用一个简单的基于堆栈的虚拟机来记录用于重建对象的指令。这是个跨版本并可自定义定义的序列化格式，但并不安全（不能防止错误或恶意资料）。错误格式或蓄意构建的资料，可能导致序列反解器导入任意模块，而且实例化任何对象。

这个函数库有另外包括序列化为标准资料格式的模块：json（内置的基本标量与集合类型支持，且能够通过编解码支持任何类型）和XML编码的属性列表（`plistlib`），限于plist支持的类型（数字，字符串，布尔，元组，串列，字典，日期时间和二进制blob）。最后，建议在正确的环境中评估对象的`__repr__`，使其和Common Lisp的打印对象大略地相符合。并非所有对象类型可以自动浸渍，特别是那些拥有操作系统资源（如文件把柄）的，但开发人员能注册自定义定义的“缩减”和构造功能，来支持任何类型的浸渍和序列化。

Pickle最初是纯粹以Python编程语言来实现的模块，但在Python 3之前的版本中，cPickle模块（也是内置的）提供了更快速的性能。cPickle从Unladen Swallow项目改造而成。在Python 3中，开发人员应该导入标准版本，该版本会尝试导入加速版本并返回纯Python版本。

.NET Framework

.NET框架有几个由微软设计的序列化器。第三方协力厂商也有许多序列化器。

Delphi

Delphi提供将组件（也称为持续对象）序列化的内置机制，完全与开发环境集成。组件的内容会被保存在DFM文件中，并即时重新加载。

OCaml

OCaml的标准函数库提供`Marshal`模块和`Pervasives`函数，`output_value`和`input_value`用于编组。虽然OCaml编程是静态类型检查的，但Marshal模块的使用可能会破坏类型保证，因为没有方法能检查反序列的流，是否代表期望类型的对象。OCaml中的函数或含有函数的数据结构（例如带有方法的对象），由于其中的执行码不可以在相异程序之间传输，所以难以将函数编组。（有一个旗标可标示函数代码的位置，但只能在完全相同的程序中解组）。标准编组功能可以配置一个旗标，来共享和循环资料的处理。

Smalltalk

通常，非递归和非共享的对象能利用`storeOn:/readFrom:`协议，以人类可读的形式来存储和截取。`storeOn:`方法产生一个Smalltalk表达式原文，而以`readFrom:`评估时：重新建立原始对象。这方案特殊之处在于它利用对象的程序描述，而不是资料本身。因此它非常有弹性，允许更紧密的表示类定义。不过在其原始形式中，它不处理循环的数据结构，也不保留共享引用的识别（即两个引用对应到单一对象，将被恢复为两个相等的引用，但这两份是不同的副本）。

为此，存在各种可携和非便携式的代替方案。其中一些属于特定的Smalltalk实现或是类库。在Squeak Smalltalk中有几种方法可以序列化和存储对象。最简单和最常用的是`storeOn:/readFrom:`，和根基于`SmartRefStream`二进制单元格式的序列化程序。此外对于包裹对象，可以用ImageSegments来存储和截取。两者都提供了所谓的“二进制对象仓存框架”，可对紧密二的进制形式执行序列化和截取。两者都处理循环的、递归的和共享的结构，存储/截取类别和父类信息，并且包括用于“即时”迁移对象的机制（将旧版编写的实例，依照不同对象布局转换成类别）。

这些API（storeBinary/readBinary）虽然彼此相似，但编码细节是不同的，使得这两种格式并不兼容。而Smalltalk/X是自由开放源码的，能被加载到其它Smalltalks方言中，允许它们之间能互相交换。对象序列化并非ANSI Smalltalk规范的一部分。因此，序列化对象的代码因Smalltalk实现而异，所得到的二进制资料也不同。例如在Ambrai中就无法恢复在Squeak中所建立的序列化对象。所以，不同Smalltalk实现的各种应用程序，无法在不同实现之间共享资料。这些应用程序包括MinneStore对象数据库和一些RPC包。这个问题的解决方案是SIXX，它是一个使用XML格式进行序列化的Smalltalks的软件包。

参考文献

外部链接