面试官：详细说说你对序列化的理解

敖丙 2022-02-22 14:08

以下文章来源于爱笑的架构师 ，作者雷小帅

爱笑的架构师
华中科技大学硕士，大厂打工中.....

凡事都要问为什么，在讲解序列化概念和原理前，我们先来了解一下为什么需要序列化。

为什么要序列化？

如果光看定义我想你很难一下子理解序列化的意义，那么我们可以从另一个角度来感受一下什么是序列化。

都玩过游戏么？玩过的同学应该知道游戏里有一个叫『存档』的功能，每次不想玩的时候可以把当前进度存档，下次有时间想玩的时候，直接载入存档就可以接着玩了，这样的好处是之前的游戏进度不会丢失，要是每次打开都重新玩估计大家也没什么耐心了。

如果把面向对象的思想带到游戏的世界，那在我们眼中不管是游戏角色还是游戏中的怪兽、装备等等都可以看成是一个个对象：

- 角色对象（包含性别、等级、经验值、血量、伤害值、护甲值等属性）
- 怪兽对象（包含类型、血量、等级等等属性）
- 装备对象（包含类型、伤害值、附加值等等属性）

在玩游戏的过程中创建一个游戏角色就好像是创建了一个角色对象，拿到一套装备就好像创建了一个装备对象，路上遇到的怪兽等等也都是对象了。

我们再用计算机的思维去思考，创建的这些对象都是保存在内存中的，大家都知道内存的数据是短暂保留的，断电之后是会消失的，但是游戏经过手动存档之后就算你关机几天了，再次进入游戏读取存档，你会发现之前在游戏中创建的角色和装备都还在呢，这就很奇怪了，明明内存的数据断电就消失了，这是为什么？

稍加思考就知道，我们在存档的过程中就是将内存中的数据存储到电脑的硬盘中，硬盘的数据在关机断电后是不会丢失的（别杠，硬盘损坏数据丢失先不考虑）。这个过程就是对象的持久化，也就是我们今天要讲的**对象序列化**。对象的序列化逆过程就叫做**反序列化**，反序列化也很好理解就是将硬盘中的信息读取出来形成对象。

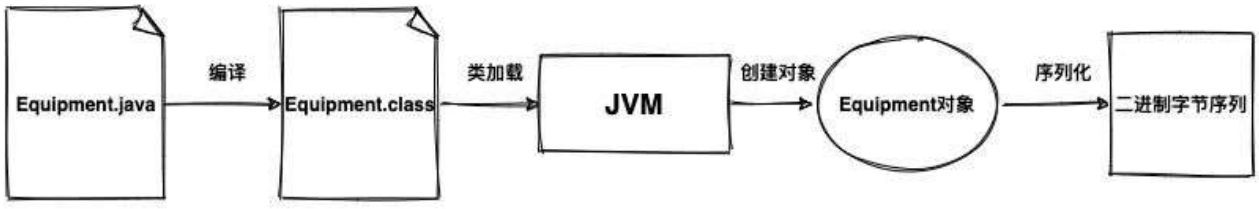
什么是序列化？

前面引入游戏的例子是为了让大家生动地理解什么是序列化和反序列化。简单总结一下就是：

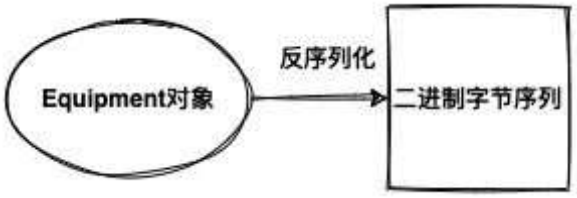
- **序列化**是指将对象实例的状态存到存储媒体的过程
- **反序列化**是指将存储在存储媒体中的对象状态装换成对象的过程

用更为抽象的概念来讲：

序列化：把对象转化为可传输的字节序列过程



反序列化：把字节序列还原为对象的过程



序列化的机制

序列化最终的目的是为了对象可以**跨平台存储**和**进行网络传输**，而我们进行跨平台存储和网络传输的方式就是**IO**，而IO支持的数据格式就是**字节数组**。

那现在的问题就是如何把对象转换成字节数组？这个很好办，一般的编程语言都有这个能力，可以很容易将对象转成字节数组。

仔细一想，我们单方面的把对象转成字节数组还不行，因为没有规则的字节数组我们是没办法把对象的本来面目还原回来的，简单说就是将对象转成字节数组容易但是将字节数组还原成对象就难了，所以我们必须在把对象转成字节数组的时候就制定一种规则（序列化），**那么我们从IO流里面读出数据的时候再以这种规则把对象还原回来**（反序列化）。

还是拿上面游戏那个例子，我们将正在玩的游戏存档到硬盘，**序列化**就是将一个个角色对象和装备对象存储到硬盘，然后留下一张原来对象的结构图纸，**反序列化**就是将硬盘里一个个对象读出来照着图纸逐个还原恢复。

常见序列化的方式

序列化只是定义了拆解对象的具体规则，那这种规则肯定也是多种多样的，比如现在常见的序列化方式有：JDK原生、JSON、ProtoBuf、Hessian、Kryo等。

（1）JDK原生

作为一个成熟的编程语言，JDK自带了序列化方法。只需要类实现了 `Serializable` 接口，就可以通过 `ObjectOutputStream` 类将对象变成byte[]字节数组。

JDK序列化会把对象类的描述信息和所有的属性以及继承的元数据都序列化为字节流，所以会导致生成的字节流相对比较大。

另外，这种序列化方式是JDK自带的，因此不支持跨语言。

简单总结一下：JDK原生的序列化方式生成的字节流比较大，也不支持跨语言，因此在实际项目和框架中用的都比较少。

（2）ProtoBuf

谷歌推出的，是一种语言无关、平台无关、可扩展的序列化结构数据的方法，它可用于通信协议、数据存储等。序列化后体积小，一般用于对传输性能有较高要求的系统。

（4）Hessian

Hessian是一个轻量级的二进制web service协议，主要用于传输二进制数据。

在传输数据前 Hessian 支持将对象序列化成二进制流，相对于 JDK 原生序列化，Hessian序列化之后体积更小，性能更优。

（5）Kryo

Kryo 是一个 Java 序列化框架，号称 Java 最快的序列化框架。Kryo 在序列化速度上很有优势，底层依赖于字节码生成机制。

由于只能限定在 JVM 语言上，所以 Kryo 不支持跨语言使用。

（6）JSON

上面讲的几种序列化方式都是直接将对象变成二进制，也就是byte[]字节数组，这些方式都可以叫二进制方式。

JSON 序列化方式生成的是一串有规则的字符串，在可读性上要优于上面几种方式，但是在体积上就没什么优势了。

另外 JSON 是有规则的字符串，不跟任何编程语言绑定，天然上就具备了跨平台。

总结一下：JSON 可读性强，支持跨平台，体积稍微逊色。

JSON 序列化常见的框架有：

`fastJSON` 、 `Jackson` 、 `Gson` 等。

序列化技术的选型

上面列举的这些序列化技术各有优缺点，不能简单地说哪一种就是最好的，不然也不会有这么多序列化技术共存了。

既然有这么多序列化技术可供选择，那在实际项目中如何选型呢？

我认为需要结合具体的项目来看，比较技术是服务于业务的。你可以从下面这几个因素来考虑：

（1）协议是否支持跨平台

如果一个大的系统有好多种语言进行混合开发，那么就肯定不适合用有语言局限性的序列化协议，比如 JDK 原生、Kryo 这些只能用在 Java 语言范围下，你用 JDK 原生方式进行序列化，用其他语言是无法反序列化的。

（2）序列化的速度

如果序列化的频率非常高，那么选择序列化速度快的协议会为你的系统性能提升不少。

（3）序列化生成的体积

如果频繁的在网络中传输的数据那就需要数据越小越好，小的数据传输快，也不占带宽，也能整体提升系统的性能，因此序列化生成的体积就很关键了。

小结

（1）为什么我们要序列化？

因为我们需要将内存中的对象存储到媒介中，或者我们需要将一个对象通过网络传输到另外一个系统中。

（2）什么是序列化？

序列化就是把对象转化为可传输的字节序列过程；**反序列化**就是把字节序列还原为对象的过程。

（3）序列化的机制

序列化最终的目的是为了对象可以**跨平台存储**和**进行网络传输**，而我们进行跨平台存储和网络传输的方式就是**IO**，而IO支持的数据格式就是**字节数组**。

将对象转成字节数组的时候需要制定一种规则，这种规则就是序列化机制。

（4）常见序列化的方式

现在常见的序列化方式有：JDK原生、JSON、ProtoBuf、Hessian、Kryo等。

（5）序列化技术的选型

选型最重要的就是要考虑这三个方面：**协议是否支持跨平台、序列化的速度、序列化生成的体积**。

阅读 2607

分享　　收藏　　7　　在看

写下你的留言

精选留言

少虞　　4

简单点，
投我以木瓜，报之以琼琚。

木瓜和琼琚路上容易坏，我拿篮子给你装上。

春