序列化

C-2 创建: 张林伟, 最后修改: 张林伟 2018-09-26 20:53

serialVersionUID

> serialVersionUID 用来表明类的不同版本间的兼容性
>
> 简单来说，Java 的序列化机制是通过在运行时判断类的 serialVersionUID 来验证版本一致性的。在进行反序列化时，JVM 会把传来的字节流中的 serialVersionUID 与本地相应实体（类）的 serialVersionUID 进行比较，如果相同就认为是一致的，可以进行反序列化，否则就会出现序列化版本不一致的异常。
>
> 当实现 java.io.Serializable 接口的实体（类）没有显式地定义一个名为 serialVersionUID、类型为 long 的变量时，Java 序列化机制会根据编译的 class 自动生成一个 serialVersionUID 作序列化版本比较用，这种情况下，只有同一次编译生成的 class 才会生成相同的 serialVersionUID 。
>
> 如果我们不希望通过编译来强制划分软件版本，即实现序列化接口的实体能够兼容先前版本，未作更改的类，就需要显式地定义一个名为serialVersionUID，类型为long的变量，不修改这个变量值的序列化实体都可以相互进行串行化和反串行化。

静态变量

序列化保存的是对象的状态，静态变量属于类的状态，因此**序列化并不保存静态变量，只序列化它的成员变量。**

transient

当某个字段被声明为transient后，默认序列化机制就会忽略该字段。

敏感字段加密

- 使用原始 ObjectOutputStream 和 ObjectInputStream 来序列化

  序列化对象添加 writeObject(ObjectOutputStream) 和 readObject(ObjectInputStream) 方法

  > 在序列化过程中，虚拟机会试图调用对象类里的 writeObject 和 readObject 方法，进行用户自定义的序列化和反序列化，如果没有这样的方法，则默认调用是 ObjectOutputStream 的 defaultWriteObject 方法以及 ObjectInputStream 的 defaultReadObject 方法。用户自定义的 writeObject 和 readObject 方法可以允许用户控制序列化的过程，比如可以在序列化的过程中动态改变序列化的数值。基于这个原理，可以在实际应用中得到使用，用于敏感字段的加密工作。

- 使用第三方工具（如jackson）

  自定义序列化方式

序列化存储规则

Java 序列化机制为了节省磁盘空间，具有特定的存储规则，当写入文件的为同一对象时，并不会再将对象的内容进行存储，而只是再次存储一份引用。

参考资料：

https://blog.csdn.net/hulefei29/article/details/2823221

https://www.ibm.com/developerworks/cn/java/j-lo-serial/index.html