**####Java基础-序列化（第七周）**

学习内容：  
1. java原生序列化方式  
2. 目前常用的序列化方案Json、Thrift、Protobuf、Kryo

任务：  
1. 各种序列化方案的对比，例如时间、空间维度

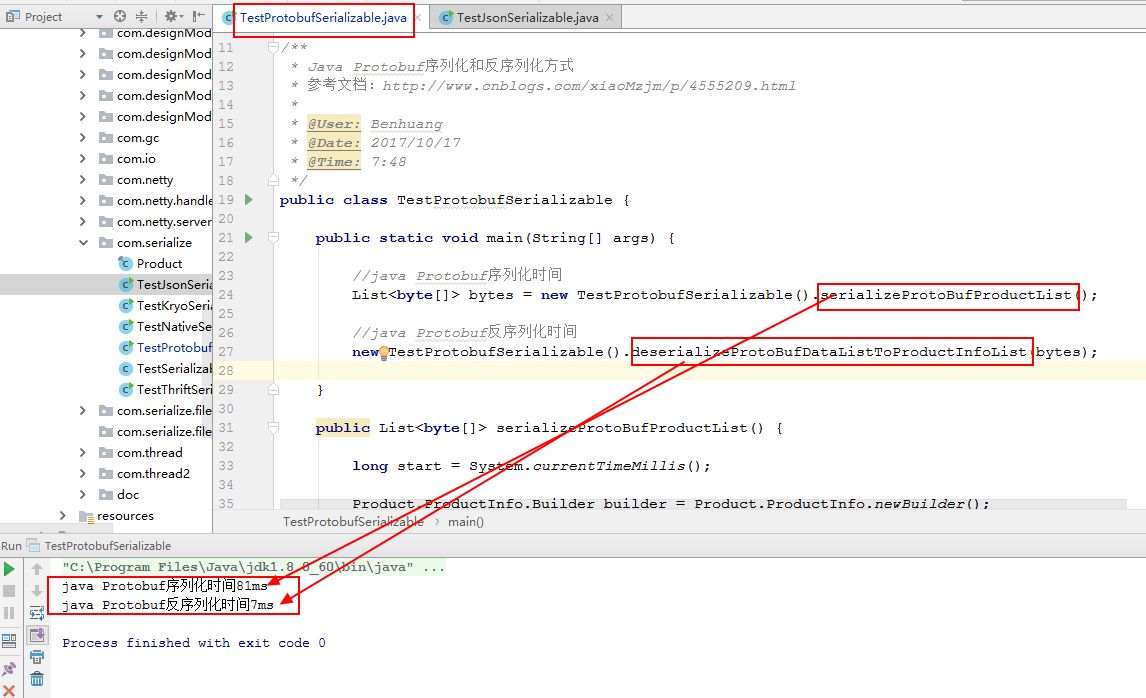
**序列化**

各种序列化方式耗时比较：**循环配置数据量10000的map**

|  |  |  |
| --- | --- | --- |
| **方式** | **序列化耗时** | **反序列化耗时** |
| 原生 | 534ms | 382ms |
| Json | 250ms | 51ms |
| Thrift | 150ms | 58ms |
| Protobuf | 82ms | 6ms |
| Kryo | 90ms | 51ms |

测试案例地址：<https://github.com/ben201708/learn_java/tree/master/src/main/java/com/serialize>

Protobuf测试情况如下图所示：



|  |  |  |
| --- | --- | --- |
| **序列化方式** | **适用语言** | **备注** |
| Kryo | Java |  |
| FST | Java |  |
| Protostuff | 跨语言 | ProtoBuf的改良版，可直接将java object进行序列化，无需像protobuf的中间转化过程 |
| ProtoBuf | 跨语言 |  |
| Thrift | 跨语言 |  |
| Avro | 跨语言 |  |
| MsgPack | 跨语言 |  |

如果对空间没有极其苛刻的要求，protostuff也许是最佳选择。protostuff支持在序列化之后，反序列化之前这段时间内，java class增加字段。但是protostuff只要保证新字段添加在类的最后，而且用的是sun系列的JDK, 是可以正常使用的。因此，如果序列化是用在缓存等场景下，序列化对象需要存储很久，可以选择protostuff。

## Java原生序列化工具

## Java本身提供的序列化工具基本上能胜任大多数场景下的序列化任务，Java自带的序列化工具在序列化过程中不仅需要将对象的完整的class name记录下来，还需要把该类的定义也都记录下，包括所有其他引用的类，这会是一笔很大的开销，尤其是仅仅序列化单个对象的时候。正因为java序列化机制会把所有meta-data记录下来，因此当修改了类的所在的包名后，反序列化则会报错。Java自带序列化工具的性能问题总结如下：

## 一个single object的序列化会递归地，连同所有成员变量（instsnce variables）一起序列化了，这种默认机制很容易造成不必要的序列化开销。

## 序列化和反序列化过程需要上面的这种机制去递归并用反射机制去寻找所有成员变量的信息，另外如果没定义自己serialVersionUID的话，那么对象及其他变量都必须自己产生一个。上述过程开销很大。

## 使用默认序列化机制，所有序列化类定义完整信息都会被记录下来，包括所有包名、父类信息、以及成员变量

## Kryo

## kryo根据上述Java原生序列化机制的一些问题，做了很多优化工作，而且提供了很多serializer，甚至封装了Unsafe类型的序列化方式，需要注意的是，jdk1.7以后，默认关闭unsafe的类（sun.misc.Unsafe）包。其中CompatibeFieldSerializer就是默认提供的一系列serializer的一种，顾名思义就是一种成员变量上下兼容的序列化工具，支持该类对成员变量的增删。另外kryo更新比较活跃，问题修复很快。

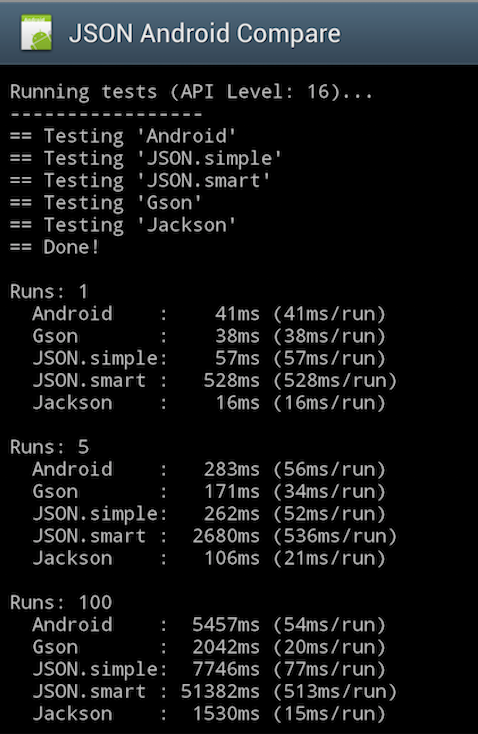
## Fast-serialization

## fst-serialozation相对来说是一个很新的序列化工具，虽然速度于kryo有一些差距，但效果几乎于kryo一致，都能瞬间反序列化出内容并渲染。

## 如果原来系统使用的Java原生的序列化工具，替换成fast-serialization非常简单：只要把Java的ObjectOutputStream与ObjectInputStream替换成FSTObjectOutput和FSTObjectInput就行了。

## JSON

## 从多种序列化工作的表现来看，比较优秀的JSON解析工具的表现还是比较好的，有些json解析工具甚至速度超过了一些二进制的序列化方式。Android环境下也有评测json解析性能的demo，图3-1，可以看出jackson在速度上还是比较有优势的，但与Android自带的json包，也没有数量级以上的优势，而jackson的jar包大小达1mb多，因此对于普通的android应用来说是比较奢侈的。



## Protocol-Buffer

## Protocol buffers是一个用来序列化结构化数据的技术，支持多种语言诸如C++、Java以及Python语言，可以使用该技术来持久化数据或者序列化成网络传输的数据。相比较一些其他的XML技术而言，该技术的一个明显特点就是更加节省空间（以二进制流存储）、速度更快以及更加灵活。 通常，编写一个protocol buffers应用需要经历如下三步：

## 1、定义消息格式文件，最好以proto作为后缀名。注意：optional、required、repeated作用，使用不当会抛异常

## 文件案例：<https://github.com/ben201708/learn_java/blob/master/src/main/java/com/serialize/file/product.proto>

## 2、使用Google提供的protocol buffers编译器来生成代码文件，一般为.h和.cc文件，主要是对消息格式以特定的语言方式描述

## Win环境编译器压缩包：<https://github.com/ben201708/learn_java/tree/master/lib/serializable/protoc.rar>

## 编译方式：protoc.exe --java\_out=./Product.proto（proto文件名），文件内容如下：

## 

## 3、使用protocol buffers库提供的API来编写应用程序