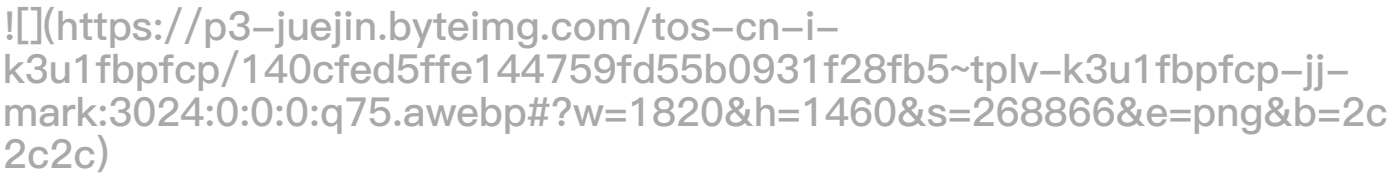
ly三个类，Family是在OldFamily中增加了新属性oldPerson。主要是为了模拟数据处理时，新老数据的兼容问题。

```
@Data
@NoArgsConstructor
@AllArgsConstructor
@ToString
public class Person {
    private String name;
    private int age;
    private int sex;
}
```

```
@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class Family {
    private Person yongPerson;
    private Person oldPerson;
    private List<Person> persons;
}
```

```
@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class OldFamily {

    private Person yongPerson;

    private List<Person> persons;
}
```

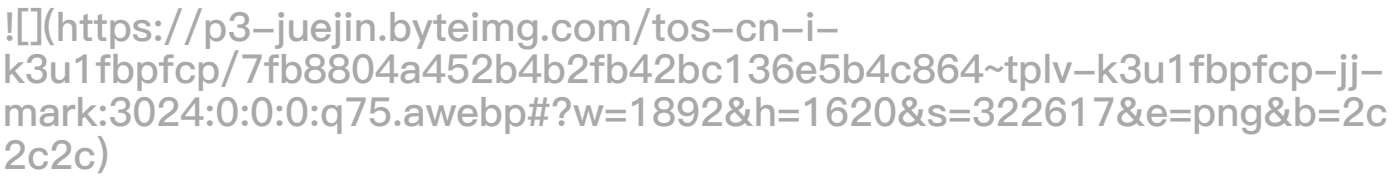
```

### fastjson序列化与反序列化



通过程序运行结果，可以观察到在Family序列化结果中，persons属性中的第三个person实例是oldPerson实例的引用，而yongPerson属性的值是persons属性中下标为0的实例的引用。这已经不是标准的JSON格式，而是fastjson的特性。当增加oldPerson属性后，Family序列化的结果在反序列化为OldFamily对象实例时，persons属性中有一个person实例为空。虽然反序列化不会报错，但程序将无法得到预期的结果。

为解决这个问题，fastjson在序列化时是默认的顺序是按照属性字段的字母顺序排序。你也可以通过注解的方式指定顺序，将新增加的属性放到orders的最后。

```
@JSONType(orders={"yongPerson", "persons", "oldPerson"})
public class Family {

    private Person yongPerson;

    private List<Person> persons;

    private Person oldPerson;

}
```
```

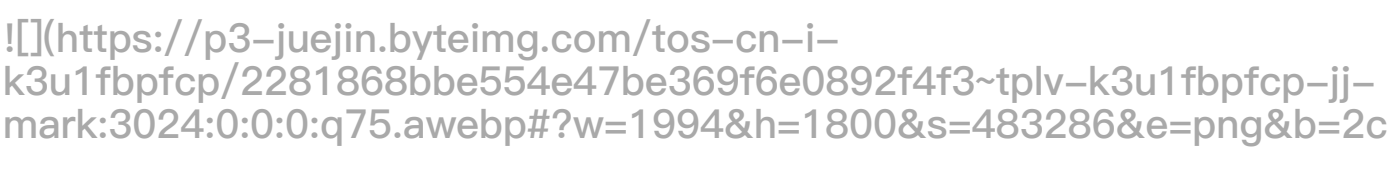


通过序列化结果可以看到oldPerson这个新增加的属性已经引用了persons属性中下标为2的实例。反序列化之后的OldFamily对象persons属性中的三个实例都有值，不会再有null值。程序还可以确保正确运行。

fastjson1.X 可以通过SerializerFeature参数来输出标准JSON格式

```
public static String toJSONString(Object object, SerializerFea
```

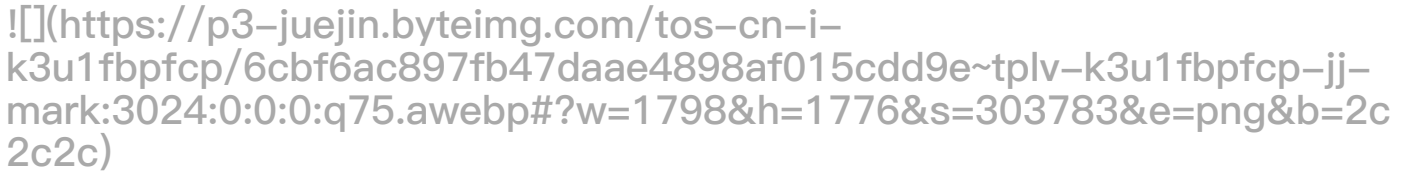
通过结果可以看出，使用jackson进行反序列化时，没有指定了DeserializationFeature.FAIL_ON_UNKNOWN_PROPERTIES为false，那么默认值为true，会把“$ref”作为属性进行检测，无论是OldPerson还是Person都没有此属性。会抛出异常。



2c2c)

通过结果可以看出，使用jackson进行反序列化时，指定了DeserializationFeature.FAIL_ON_UNKNOWN_PROPERTIES为false没有报异常，但是person对象是以属性的默认值实例化的，没有得到想要的结果。

fastjson默认序列化gson反序列化



通过结果可以看出，使用gson进行反序列化时没有报异常，但是person对象是以属性的默认值实例化的，没有得到预期结果。

总结：

1.使用fastjson时，默认的序列化方式会对于具有相同对象的多个引用，除了第一个会以标准的JSON文本输出，其他引用会以“$ref”的方式输出文本。为了以标准的JSON格式输出文本，可以使用`SerializerFeature.DisableCircularReferenceDetect`参数。而fastjson2的默认序列化输出是标准的JSON格式，若需要具有fastjson默认序列化特性的场景，可以指定`com.alibaba.fastjson2.JSONWriter.Feature`参数值为`JSONWriter.Feature.ReferenceDetection`。

2.在使用fastjson或者fastjson2的特性，具有相同对象的多个引用，除了第一个会以标准的JSON文本输出，其他引用会以“$ref”的方式输出文本。在新增类

属性字段的情况下，一定要把新增的属性放到最后序列化，确保在反序列化成没有新增属性类实例时，得不到预期结果造成线上事故。

3.fastjson和fastjson2在一定程度上是兼容的，但也存在版本差异和Bug，因此在使用时需要进行充分的测试验证。另外，gson、jackson和fastjson2的默认序列化方式也是标准JSON格式。对于jackson，若要在反序列化时兼容不存在的属性的JSON文本成为对象实例，需要设定`DeserializationFeature.FAIL_ON_UNKNOWN_PROPERTIES`为false。

4.在使用JSON序列化工具时，需要注意每个工具的特点，尤其是在使用各自具有的特性时，要留意兼容性问题。在进行JSON工具版本升级时，也要进行充分的测试验证，确保有单元测试来保证质量。

5.若作为与外部系统交互的JSON格式数据，需要以标准化的数据格式进行存储或传输。

6.以上内容仅代表个人观点和一小部分实践，欢迎大家一起探讨。

作者：京东物流 吴宪彬

来源：京东云开发者社区

原文链接: https://juejin.cn/post/7370901494595731471