后知、后觉

耐得住寂寞 ,才能享受繁华。

理解serialVersionUID是什么？有什么用？如何生成？

如果您曾经实现过Serializable接口，则必须遇到此警告消息

```
The serializable class xxx does not declare a static final serialVersionUID field of type long
```

那么.....什么是serialVersionUID?

serialVersionUID用作Serializable类中的版本控件。如果您没有显式声明serialVersionUID，JVM将根据您的Serializable类的各个方面自动为您执行此操作，如Java（TM）对象序列化规范中所述。

1. SerialVersionUID示例

上面的语句在开头有点难以理解（至少我做过），让我们开始一个例子来了解Serializable类如何使用SerialVersionUID来实现版本控制。

1.1 Address.java

serialVersionUID为1L的可序列化类。

```
import java.io.Serializable;

public class Address implements Serializable{

	private static final long serialVersionUID = 1L;

	String street;
	String country;

	public void setStreet(String street){
	    this.street = street;
	}

	public void setCountry(String country){
	    this.country = country;
	}

	public String getStreet(){
	    return this.street;
	}

	public String getCountry(){
	    return this.country;
	}

	@Override
	public String toString() {
	    return new StringBuffer(" Street : ")
```

计算机网络(7)
蓝桥杯练习 (4)
面经(3)
人脸识别(6)
软件设计师考试（中级） (1)
设计模式(4)
深入理解java并发(11)
更多

随笔档案

2021年1月(1)
2020年9月(1)
2020年8月(7)
2020年7月(20)
2019年10月(3)
2019年9月(2)
2019年8月(3)
2019年7月(4)
2019年6月(3)
2019年4月(4)
2019年3月(8)
2018年11月(10)
2018年10月(22)
2018年9月(34)
2018年8月(28)
更多

相册

背景(7)
背景1(10)

阅读排行榜

1. linux常用命令（50个）(491990)
2. 深入理解token(198477)
3. 理解serialVersionUID是什么？有什么用？如何生成？(39224)
4. log4j配置详解(非常详细)(33241)
5. java面试题之----转发（forward）和重定向（redirect）的区别(24226)

评论排行榜

1. 深入理解token(16)
2. 红黑树（R-B Tree）(5)
3. 深入理解http协议的特点(3)
4. linux常用命令（50个）(3)
5. java面试题之----String的intern(3)

推荐排行榜

1. linux常用命令（50个）(49)
2. 深入理解token(44)
3. java面试题之----转发（forward）和重定向（redirect）的区别(8)
4. 深入理解java的形参和实参(6)
5. java面试题之----get和post请求方法的区别(4)

最新评论

1. Re:java中什么是上下文（servlet Context)
so,dude, why is 老王
--CharltonW
2. Re:Java并发--Java中的CAS操作和实现原理
通透

```
            .append(this.street)
            .append(" Country : ")
            .append(this.country).toString();
        }
}
```

1.2 WriteObject.java

将Address对象写入/序列化为文件的简单类 – “c：\\ address.ser”。

```
import java.io.FileOutputStream;
import java.io.ObjectOutputStream;

public class WriteObject{

    public static void main (String args[]) {

        Address address = new Address();
        address.setStreet("wall street");
        address.setCountry("united states");

        try{

         FileOutputStream fout = new FileOutputStream("c:\\address.ser");
         ObjectOutputStream oos = new ObjectOutputStream(fout);
         oos.writeObject(address);
         oos.close();
         System.out.println("Done");

        }catch(Exception ex){
            ex.printStackTrace();
        }
    }
}
```

1.3 ReadObject.java

从文件中读取/反序列化Address对象的简单类 – “c：\\ address.ser”。

```
import java.io.FileInputStream;
import java.io.ObjectInputStream;

public class ReadObject{

   public static void main (String args[]) {

        Address address;

        try{
```

```
            FileInputStream fin = new FileInputStream("c:\\address.ser");
            ObjectInputStream ois = new ObjectInputStream(fin);
            address = (Address) ois.readObject();
            ois.close();

            System.out.println(address);

        }catch(Exception ex){
            ex.printStackTrace();
        }
    }
}
```

2.测试

让我们做一些测试来演示serialVersionUID的使用。

2.1相同的serialVersionUID

相同的serialVersionUID，在反序列化过程中没有问题

```
javac Address.java javac WriteObject.java javac ReadObject.java java WriteObject
java ReadObject Street : wall street Country : united states
```

复制

2.2不同的serialVersionUID

在Address.java中，**将serialVersionUID更改为2L（它是1L），然后再次编译它。**

```
javac Address.java java ReadObject java.io.InvalidClassException: Address; local
class incompatible: stream classdesc serialVersionUID = 1, local class
serialVersionUID = 2 ... at ReadObject.main(ReadObject.java:14)
```

“ InvalidClassException”会引发，因为您使用**serialVersionUID“1L”编写了一个序列化类，但尝试使用更新的序列化类serialVersionUID“2L”将其检索回来。**

serialVersionUID必须在序列化和反序列化过程中匹配。

什么时候应该更新你的serialVersionUID?

如果使用对可序列化类的某些不兼容的Java类型更改更新序列化类，则必须更新serialVersionUID。

有关可序列化类的兼容和不兼容Java类型更改的详细信息，请参阅Java对象序列化规范。

3.默认的serialVersionUID有什么问题?

如果没有声明serialVersionUID，JVM将使用自己的算法生成默认的SerialVersionUID，您可以在此处检查算法。

默认的serialVersionUID计算对类详细信息非常敏感，可能因不同的JVM实现而异，并且在反序列化过程中会导致意外的InvalidClassExceptions。

3.1客户端/服务器环境

– 客户端在Windows中使用SUN的JVM。

– 服务器在Linux中使用JRockit。

客户端通过套接字向服务器发送带有默认生成的serialVersionUID（例如123L）的可序列化类，服务器可以在反序列化过程中生成不同的serialVersionUID（例如124L），并引发意外的InvalidClassExceptions。

3.2文件/数据库环境

– App＃1在Windows中使用SUN的JVM。

– App＃2在Linux中使用JRockit。

序列化允许保存到文件或数据库中。App＃1默认生成serialVersionUID（例如123L）将可序列化类存储到数据库中，而App＃2可能在反序列化过程中生成不同的serialVersionUID（例如124L），并引发意外的InvalidClassExceptions。

您可以在此处查看JVM实现的列表。

4.如何生成serialVersionUID

您可以使用JDK“ serialver”或Eclipse IDE自动生成serialVersionUID，详见详细信息。

结论

SUN强烈建议开发人员声明serialVersionUID以避免上面列出的不同JVM问题，但我建议您应该了解什么是序列化，serialVersionUID如何实现版本控制以及您的类需要使用序列化的原因。了解serialVersionUID概念优于任何推荐的盲目。

本文参考：https://www.mkyong.com/java-best-practices/understand-the-serialversionuid/

参考

1. http://en.wikipedia.org/wiki/List_of_JVM_implementations
2. http://java.sun.com/javase/6/docs/platform/serialization/spec/class.html#4100
3. http://stackoverflow.com/questions/419796/explicit-serialversionuid-considered-harmful
4. http://en.wikipedia.org/wiki/Serialization#Java
5. http://www.javaworld.com/javaworld/jw-02-2006/jw-0227-control.html?page=1
6. http://www.javablogging.com/what-is-serialversionuid/
7. http://java.dzone.com/articles/dont-ignore-serialversionuid
8. http://www.java-forums.org/new-java/8196-serialversionuid.html

分类: java SE

后知、后觉
粉丝 – 126 关注 – 14

« 上一篇： Java并发--Java中的CAS操作和实现原理

» 下一篇： java1.8新特性整理（全）

posted @ 2019-08-22 15:01 后知、后觉 阅读(39224) 评论(0) 编辑 收藏 举报