ited

ing language, FIX Protocol, Tibco Rendezvous and related Java technology stack.

Search

FRIDAY, JANUARY 20, 2012

Difference between Serializable and Externalizable in Java Serialization

Difference between serializable and externalizable is popular java interview question which we have touched on my earlier post on Serialization: Top 10 java serialization interview questions. knowing *differences between externalizable and serializable* is not just important from interview point of view but also getting control of serialization process and optimizing performance of serialization. both **serializable** and **extenalizable** used to serialize or persist java objects but the way they do is little different. In case of Serializable Java Virtual machine has full control for serializing object while in case of Externalizable, application gets control for persisting objects. `writeExternal()` and `readExternal()` method provides complete **control on format and content of Serialization process** to application which can be leverage to increase performance and speed of serialization process.

Serialization and Externalization in Java

Serializable vs Externalization in Java

here are some more differences between Serializable and Externalizable interface in Java:

1. In case of Serializable, **default serialization process** is used. while in case of Externalizable custom Serialization process is used which is implemented by application.

2. JVM gives call back to `readExternel()` and `writeExternal()` of `java.io.Externalizalbe` interface for restoring and writing objects into persistence.

3. **Externalizable** interface provides complete control of serialization process to application.

4. `readExternal()` and `writeExternal()` supersede any specific implementation of `writeObject` and `readObject` methods.

Though Externalizable provides complete control, it also presents challenges to serialize super type state and take care of default values in case of transient variable and static variables in Java. If used correctly **Externalizable** interface can improve performance of serialization process.

That's all on **Difference between Externalizable and Serializable interface in Java**. This is always asked when Java interview take turn towards Serialization after Multi-Threading questions and Collections Interview questions. Only problem with Serialization is that not many programmer use it and that's why it look little difficult otherwise once you familiar with Serialization process and rules of Serialization, interview questions can be more easily handled.

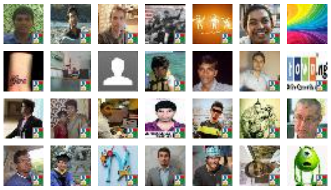
Some Interview Questions post:

Top 30 UNIX Command Interview Questions Answers
Top 30 Programming Interview Questions
10 Interview questions on Singleton Pattern in Java
10 Spring Interview Questions Answers for J2EE developers
Top 10 Struts Interview Questions in Java
Top 10 Singleton Pattern Interview questions in Java
10 JSP Interview Questions and Answers
10 Servlet Interview questions and Answers

Followers

Join this site
with Google Friend Connect

Members (2734) More »



Already a member? Sign in

Follow Us

Follow @javinpaul

Javarevisited

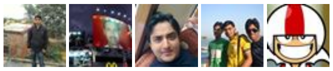
Follow +1

+ 44,752

Javarevisited

Like

16,589 people like Javarevisited.



Facebook social plugin

Subscribe by email:

Subscribe

By Javin Paul

Blog Archive

- ► 2015 (13)
- ► 2014 (106)
- ► 2013 (136)
- ▼ 2012 (217)
  - ► December (52)
  - ► November (8)
  - ► October (14)
  - ► September (8)
  - ► August (9)
  - ► July (9)
  - ► June (12)
  - ► May (10)
  - ► April (14)
  - ► March (28)
  - ► February (18)
  - ▼ January (35)
    - How to reverse String in Java using Iteration and ...
    - Sudden Drop in Traffic? Could be Google page layou...
    - How to get and set default Character encoding or C...
    - How to define Error page in Java Web Application -...
    - How to fix java.sql.SQLException: Invalid column

You might like:

- 6 Books to Learn and Master Programming and Coding - Must Read
- Difference between char, varchar, nchar and nvarchar data type in SQL Server database
- Top 50 Java Thread Interview Questions Answers for Experienced
- How to check if two String are Anagram in Java - Program Example

Recommended by

Posted by Javin Paul at 6:31 AM

g+1 +4 Recommend this on Google

Labels: core java , core java interview question

Location: United States

7 comments :

Anonymous said...

an important difference is that Serializable will call all of the superclasses, and Externalizable will not.

February 22, 2012 at 4:42 AM

SARAL SAXENA said...

Hi Javin gr8 article ....few things that I want to add..

By implementating java.io.Serializable, you get "automatic" serialization capability for objects of your class. No need to implement any other logic, it'll just work. The Java runtime will use reflection to figure out how to marshal and unmarshal your objects.

In earlier version of Java, reflection was very slow, and so serializaing large object graphs (e.g. in client-server RMI applications) was a bit of a performance problem. To handle this situation, the java.io.Externalizable interface was provided, which is like java.io.Serializable but with custom-written mechanisms to perform the marshalling and unmarshalling functions (you need to implement readExternal and writeExternal methods on your class). This gives you the means to get around the reflection performance bottleneck.

In recent versions of Java (1.3 onwards, certainly) the performance of reflection is vastly better than it used to be, and so this is much less of a problem. I suspect you'd be hard-pressed to get a meaningful benefit from Externalizable with a modern JVM.

Also, the built-in Java serialization mechanism isn't the only one, you can get third-party replacements, such as JBoss Serialization, which is considerably quicker, and is a drop-in replacement for the default.

A big downside of Externalizable is that you have to maintain this logic yourself - if you add, remove or change a field in your class, you have to change your writeExternal/readExternal methods to account for it.

In summary, Externalizable is a relic of the Java 1.1 days. There's really no need for it any more

March 27, 2013 at 3:29 AM

Anonymous said...

@SARAL SAXENA
Did you copy-paste your comments from following ?? LOL !!!
http://stackoverflow.com/questions/817853/what-is-the-difference-between-serializable-and-externalizable-in-java

February 26, 2014 at 1:04 PM

Anonymous said...

You can generate java value objects that implement Externalizable for you without having to write the code yourself using VALJOGen (http://valjogen.41concepts.com/).

November 22, 2014 at 10:51 AM

Unknown said...

Tip: ValjoGen can generate java value objects for you that implement Externalizable (and much more) for you so you do not have to write the code manually. ValjoGen is free open source at the website is at http://valjogen.41concepts.com.

November 22, 2014 at 11:06 AM

Javin Paul said...

Thanks @Unknown, this looks like a good utility which will minimize mistakes Java developers make while using Externalizable interface.

November 22, 2014 at 6:36 PM

Rajat said...

Some few points that I came across:

i...
Difference between Thread vs Runnable interface in...
Top 10 Google Interview Questions Answers for Soft...
How to find if JVM is 32 or 64 bit from Java progr...
What is Constructor overloading in Java with Examp...
Difference between Serializable and Externalizable...
Difference between URL-rewriting URL-encoding in S...
How to get max memory, free memory and total memor...
How to use Assertion in Java Code - When, Where
How to enable or disable Assertion in Java
JDBC Performance Tips - 4 Tips to improve performa...
How to Sort Array in Java ascending and descending...
What is Assertion in Java - Java Assertion Tutoria...
How to hide file from Java Program with Code Examp...
How to check if a File is hidden in Java with Exam...
How to get current Date Timestamps in Java on GMT ...
Difference between include directive and include a...
10 Example of this keyword in Java
How to Sort Java ArrayList in Descending Order - E...
How to write Thread-Safe Code in Java
How to create and initialize Anonymous array in Ja...
How to convert ArrayList to Set in Java with Examp...
What is FIX Engine in FIX Protocol
REST Web services Framework interview questions an...
How to Change or Set File Permissions in Java – Co...
How to Use Code Point Methods of Java String get U...
10 Example of Hashtable in Java – Java Hashtable T...
Why use Memory Mapped File or MapppedByteBuffer in...
How to check File Permission in Java with Example ...
What is Marker interfaces in Java and why required...
Tomcat – java.lang.OutOfMemoryError: PermGen space...

► 2011 (145)
► 2010 (33)

References

Java API documentation JDK 6

Serialization is a recursive algorithm. What I mean to say here is, apart from the fields that are required, starting from a single object, until all the objects that can be reached from that object by following instance variables, are also serialized. This includes the super class of the object until it reaches the "Object" class and the same way the super class of the instance variables until it reaches the "Object" class of those variables. Basically all the objects that it can read. This leads to lot of overheads.

Both serializing and deserializing require the serialization mechanism to discover information about the instance it is serializing. Using the default serialization mechanism, will use reflection to discover all the field values. Also the information about class description is added to the stream which includes the descption of all the serializable superclasses, the description of the class and the instance data associated with the specific instance of the class. Lots of data and metadata and again performance issue.

You know that serialization needs serialVersionUID, a unique Id to identify the information persisted. If you dont explicitly set a serialiVersionUID, serialization will compute the serialiVersionUID by going through all the fields and methods. So based on the size of the class, again serialization mechanism takes respective amount of time to calculate the value. A third performance issue.

Externalizable interface is not a marker interface and it provides two methods - writeExternal and readExternal.

Externalization efficiency comes at a price. The default serialization mechanism adapts to application changes due to the fact that metadata is automatically extracted from the class definitions (observe the format above and you will see that when the object is serialized by implementing Serializable interface, the class metadata(definitions) are written to the persistent store while when you serialize by implementing Externalizable interface, the class metadata is not written to the persistent store). Externalization on the other hand isn't very flexible and requires you to rewrite your marshalling and demarshalling code whenever you change your class definitions.

As you know a default public no-arg constructor will be called when serializing the objects that implements Externalizable interface. Hence, Externalizable interface can't be implemented by Inner Classes in Java as all the constructors of an inner class in Java will always accept the instance of the enclosing class as a prepended parameter and therefore you can't have a no-arg constructor for an inner class. Inner classes can achieve object serialization by only implementing Serializable interface.

If you are subclassing your externalizable class, you have to invoke your superclass's implementation. So this causes overhead while you subclass your externalizable class. Observe the examples above where the superclass writeExternal method is explicitly called in the subclass writeExternal method.

Methods in externalizable interface are public. So any malicious program can invoke which results into loosing the prior serialized state.

December 18, 2014 at 5:40 AM

Post a Comment

Enter your comment...

Comment as: Select profile...

Publish **Preview**

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)

About Me Privacy Policy

Spring framework doc

Struts

JDK 7 API

MySQL

Linux

Eclipse

jQuery