Transient keyword with Serialization in Java

November 3, 2016 SJ Serialization 0

In this article, we will discuss *transient keyword or modifier* with *serialization* in detail

Whenever, we talk about *Serialization* then definitely there will be *loads of questions* on transient keyword

Also, it's one of the favorite *interview questions* in Java

Serialization process

During serialization process i.e.; saving the state of an Object to File, all *instance variables* will be *participated and persisted to file storage*

Instance variable: all member variables/attributes of an Object without any modifiers like *static*

What if we don't want to serialize specific variable/attributes for security or any other reasons?

- The answer is ***declare respective member*** instance variable with ***transient*** modifier
- Yes, we can ***stop*** persisting specific variable during serialization process by ***declaring transient modifier*** (for that specific variable)

Spring JDBC: An example on JdbcTemplate using Annotation
Java JDBC: An example to connect MS Access database
Oracle OSB 12c: Service Callout and Routing Table example
Oracle OSB 12c: Hello World service with both Business and Proxy Service

Transient keyword

- Transient keyword or modifier is ***applicable only for variables***
- We can ***stop*** persisting specific variable, ***by declaring transient keyword***
- During serialization, JVM ***ignores*** the ***original value of transient variable*** and saves ***default value*** to file
- **Examples:** Customer SSN or password ***need not to be stored***. Hence, it's a ***good practice to declare*** those variables as transient
- So whenever we encounter ***transient*** keyword, it means that ***not to serialize***

Demo example on Transient keyword

For objects to participate in serialization & de-serialization process, corresponding *class* should implement *java.io.Serializable* interface

Exception: otherwise, *run time* exception will be thrown stating *NotSerializableException*

Step 1: Create POJO which implements *java.io.Serializable* interface

In Customer POJO, there are *4 member variables* with *customerSSN* declared with *transient* keyword

Which means, during serialization instead of original value, *default value* will be saved to file and this can be proved by *de-serializing the serialized object*

Customer.java

```
package in.bench.resources.serialization;

import java.io.Serializable;

public class Customer implements Serializable {

    // member variables
    int customerId;
    String customerName;
    int customerAge;
    transient int customerSSN;


    // 4-arg parameterized constructor
    public Customer(int customerId, String cust
            int customerAge, int customerSSN) {
        super();
        this.customerId = customerId;
        this.customerName = customerName;
        this.customerAge = customerAge;
        this.customerAge = customerAge;
    }

    // overriding toString() method
    @Override
    public String toString() {
        return "Customer [customerId=" + custom
                + ", customerName=" + customerN
                + ", customerAge="  + customerA
                + ", customerSSN=" + customerSS
    }
}
```

Step 2: Main program to demonstrate serialization/de-serialization

To Serialize: any Object, we can use *ObjectOutputStream* & *FileOutputStream* to *write/save* to the *file* (in binary format)

To De-Serialize: any Object, we can use *ObjectInputStream* & *FileInputStream* to *read/restore* from *file* (which is in binary format) into Java *heap memory*

TransientDemo.java

```java
package in.bench.resources.serialization;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;

public class TransientDemo {

    public static void main(String[] args) {

        // create an customer instance using 4-
        Customer serializeCustomer =
                new Customer(102, "SR", 17, 112

        // creating output stream variables
        FileOutputStream fos = null;
        ObjectOutputStream oos = null;

        // creating input stream variables
        FileInputStream fis = null;
        ObjectInputStream ois = null;

        // creating customer object reference
        // to hold values after de-serializatio
        Customer deSerializeCustomer = null;

        try {
            // for writing or saving binary dat
            fos = new FileOutputStream("Custome

            // converting java-object to binary
            oos = new ObjectOutputStream(fos);

            // writing or saving customer objec
            oos.writeObject(serializeCustomer);
            oos.flush();
            oos.close();

            System.out.println("Serialization s
                    + "object saved to Customer

            // reading binary data
            fis = new FileInputStream("Customer

            // converting binary-data to java-o
            ois = new ObjectInputStream(fis);

            // reading object's value and casti
            deSerializeCustomer = (Customer) oi
            ois.close();

            System.out.println("De-Serializatio
                    + "object de-serialized fro
        }
        catch (FileNotFoundException fnfex) {
            fnfex.printStackTrace();
        }
        catch (IOException ioex) {
            ioex.printStackTrace();
```

```
63            }
64            catch (ClassNotFoundException ccex) {
65                ccex.printStackTrace();
66            }
67
68            // printing customer object to console
69            System.out.println("Printing customer v
70                    + "de-serialized object... \n"
71        }
72    }
```

Output:

```
1   Serialization success: Customer object saved t
2
3   De-Serialization success: Customer object de-ser
4   from Customer.ser file
5
6   Printing customer values from de-serialized obje
7   Customer [customerId=102, customerName=SR, custo
```

Explanation:

- In above Customer POJO, ***customerSSN*** declared as ***transient***
- So during serialization process, original value of customerSSN ***won't be saved*** to file
- Instead ***default value will be saved*** (i.e.; 0 for int, null for String, etc)
- ***1st half*** of the program illustrate ***serialization*** process
- And ***2nd half*** deals with ***de-serialization process***, which ***de-serializes*** the serialized Object
- While de-serializing ***all instance member*** values are ***re-stored back perfectly*** except for customerSSN
- **Reason:** because this is ***marked*** with ***transient*** keyword

So, by declaring instance variable with ***transient keyword*** we can ***restrict to store*** that particular variable into ***file*** storage during ***serialization process***

And it depends purely on ***business requirement*** that, which all the ***instance variables*** need to be ***restricted***

References:

https://docs.oracle.com/javase/7/docs/api/java/io/Serializable.html
https://docs.oracle.com/javase/7/docs/platform/serialization/spec/serial-arch.html
https://docs.oracle.com/javase/7/docs/api/java/io/ObjectOutputStream.html
https://docs.oracle.com/javase/7/docs/api/java/io/ObjectInputStream.html
https://docs.oracle.com/javase/7/docs/api/java/io/FileOutputStream.html
https://docs.oracle.com/javase/7/docs/api/java/io/FileInputStream.html
http://docs.oracle.com/javase/specs/jls/se7/html/jls-8.html#jls-8.3.1.3

Read Also:

- Java Serialization and De-Serialization Tutorial Index
- Serialization and De-Serialization in Java
- Serializable interface
- Transient keyword with static variable in Serialization
- Transient keyword with final variable in Serialization
- Serializing a variable with transient modifier or keyword
- Order of Serialization and De-Serialization
- Serialization with Aggregation
- Serialization with Inheritance
- Externalizable interface with example
- Serializable v/s Externalizable
- Importance of SerialVersionUID in Serialization
- Singleton Design pattern with Serialization
- How to stop Serialization in Java
- How to construct a singleton class in a multi-threaded environment in Java
- How to serialize and de-serialize ArrayList in Java

Happy Coding !!
Happy Learning !!

« **Transient keyword with static variable in Serialization**

Serializable interface »

Related Posts:

1. Transient keyword with static variable in Serialization
2. Transient keyword with final variable in Serialization
3. Serializing a variable with transient modifier or keyword
4. Externalizable interface with example

DESERIALIZATION JAVA SERIALIZABLE SERIALIZATION TRANSIENT