Java - Serialization

Advertisements

Previous Page Nex

Java provides a mechanism, called object serialization where an object can be rep sequence of bytes that includes the object's data as well as information about the objec types of data stored in the object.

After a serialized object has been written into a file, it can be read from the file and dese the type information and bytes that represent the object and its data can be used to recrea memory.

Most impressive is that the entire process is JVM independent, meaning an object can b one platform and deserialized on an entirely different platform.

Classes **ObjectInputStream** and **ObjectOutputStream** are high-level streams that conta for serializing and deserializing an object.

The ObjectOutputStream class contains many write methods for writing various data method in particular stands out:

```
public final void writeObject(Object x) throws IOException
```

The above method serializes an Object and sends it to the output stream. ObjectInputStream class contains the following method for deserializing an object:

```
public final Object readObject() throws IOException,
                                 ClassNotFoundException
```

This method retrieves the next Object out of the stream and deserializes it. The return val you will need to cast it to its appropriate data type.

To demonstrate how serialization works in Java, I am going to use the Employee class tha early on in the book. Suppose that we have the following Employee class, which i Serializable interface:

```
public class Employee implements java.io.Serializable
{
   public String name;
   public String address;
   public transient int SSN;
   public int number;
   public void mailCheck()
   {
      System.out.println("Mailing a check to " + name
                         + " " + address);
   }
}
```

Notice that for a class to be serialized successfully, two conditions must be met:

The class must implement the java.io.Serializable interface.

All of the fields in the class must be serializable. If a field is not serializable, it n transient.

If you are curious to know if a Java Standard Class is serializable or not, check the docum class. The test is simple: If the class implements java.io.Serializable, then it is serializable not.

Serializing an Object:

The ObjectOutputStream class is used to serialize an Object. The following Serialize instantiates an Employee object and serializes it to a file.

When the program is done executing, a file named employee.ser is created. The prc generate any output, but study the code and try to determine what the program is doing.

Note: When serializing an object to a file, the standard convention in Java is to give extension.

```
import java.io.*;

public class SerializeDemo
{
   public static void main(String [] args)
   {
      Employee e = new Employee();
      e.name = "Reyan Ali";
      e.address = "Phokka Kuan, Ambehta Peer";
      e.SSN = 11122333;
      e.number = 101;
      try
      {
         FileOutputStream fileOut =
         new FileOutputStream("/tmp/employee.ser");
         ObjectOutputStream out = new ObjectOutputStream(fileOut)
         out.writeObject(e);
         out.close();
         fileOut.close();
         System.out.printf("Serialized data is saved in /tmp/empl
      }catch(IOException i)
      {
          i.printStackTrace();
      }
   }
}
```

Deserializing an Object:

The following DeserializeDemo program deserializes the Employee object created in the program. Study the program and try to determine its output:

```
import java.io.*;
public class DeserializeDemo
{
   public static void main(String [] args)
   {
      Employee e = null;
      try
      {
         FileInputStream fileIn = new FileInputStream("/tmp/emplo
         ObjectInputStream in = new ObjectInputStream(fileIn);
```

```
         e = (Employee) in.readObject();
         in.close();
         fileIn.close();
      }catch(IOException i)
      {
         i.printStackTrace();
         return;
      }catch(ClassNotFoundException c)
      {
         System.out.println("Employee class not found");
         c.printStackTrace();
         return;
      }
      System.out.println("Deserialized Employee...");
      System.out.println("Name: " + e.name);
      System.out.println("Address: " + e.address);
      System.out.println("SSN: " + e.SSN);
      System.out.println("Number: " + e.number);
   }
}
```

This would produce the following result:

```
Deserialized Employee...
Name: Reyan Ali
Address:Phokka Kuan, Ambehta Peer
SSN: 0
Number:101
```

Here are following important points to be noted:

The try/catch block tries to catch a ClassNotFoundException, which is declared by t method. For a JVM to be able to deserialize an object, it must be able to find the b class. If the JVM can't find a class during the deserialization of an objec ClassNotFoundException.

Notice that the return value of readObject() is cast to an Employee reference.

The value of the SSN field was 11122333 when the object was serialized, but beca transient, this value was not sent to the output stream. The SSN field of the deseria object is 0.

Previous Page Print Version PDF Version Nex

Advertisements

Online Java Classes

Easy Online Java Courses. Free 10 Day Trial. Sign Up Now!

ASP.NET | jQuery | AJAX | ANT | JSP | Servlets | log4j | iBATIS | Hibernate | JDBC | Struts | HTM

Copyright © 2014 by tutorialspoint. All Rights Reserved.