Serialización

Serialization

Samir Genaim

¿Qué es Serialización?

- Escribir/leer un objeto en/desde un Stream
- Se usa para almacenar objetos en archivos y usarlos más tarde, trasmitirlos por la red, etc..
- En Java la serialización se hace usando los siguientes clases de stream:
  1. ObjectOutputStream (writeObject)
  2. ObjectInputStream (readObject)
- Se puede serializar un objeto si su clases implementa la interfaz java.io.Serializable (a maker interface). Casi todos los clases de las librerías de Java implementan java.io.Serializable.

Escribir un Objeto en un Fichero

```
public static void saveObj(String fname, Object o) throws IOException {
    FileOutputStream os = new FileOutputStream(fname);
    ObjectOutputStream oos = new ObjectOutputStream(os);
    oos.writeObject(o);
    oos.close();
}
```

crear un OutputStream al que vamos escribir el objeto, en este caso se trata de un FileOutputStream ya que queremos almacenar en un archivo

Crear un ObjectOutputStream y pasarlo el OutputStream al que se debe escribir el objeto serializado

Serializar el objeto (escribir en el OutputStream). Se escribe suficiente información para poder reconstruir de nuevo cuando sea necesario (campos, la información de clase, pero no métodos).

Cerrar el stream

Leer un Objeto desde un Fichero

```
public static Object readObject(String fname)
                        throws IOException, ClassNotFoundException {

    FileInputStream is = new FileInputStream(fname);
    ObjectInputStream ois = new ObjectInputStream(is);
    Object o = ois.readObject();
    ois.close();
    return o;
}
```

crear un InputStream desde el que vamos a leer un objeto serializado, en este caso se trata de un FileInputStream ya que queremos leer de un archivo

Crear un ObjectInputStream y pasarlo el InputStream desde el que se debe leer el objeto serializado

Cerrar el stream

Leer el objeto serializado (desde el InputStream). Devuelve una instancia de la misma clase del objeto que ha sido serializado. Se lanza ClassNotFoundException si no puede encontrar esa clase ...

Ejemplo de of Serialización

```
public class Person implements Serializable {
    private String name;
    private Integer id;
    public Person(String name, Integer id) {
        this.name = name;
        this.id = id;
    }
    @Override
    public String toString() {
        return "("+name+","+id+")";
    }
}
```

Implementar la interfaz Serializable

p y q son instancias de Person con la misma información, pero son dos objetos distintos!!

```
public static void main(String[] args) throws … {
    Person p = new Person(“John”,2);
    saveObject("/tmp/bla", p);
    Person q = (Person) readObject("/tmp/bla");
    System.out.println(p+" : "+" "+q+" : "+(p==q));
}
```

see: examples.serialazation.ex1

Atributos no Serializables

- Para serializar un objeto, todos sus atributos deben ser serializables, si no se lanza una excepción ..
- Si tenemos una clase con algunos atributos no es serializables, podemos serializarlo declarando esos atributos como transient
- Al deserializarlo esos atributos tendrán el valor null

```
public class Person implements Serializable {

    private String name;
    private Integer id;

    transient SomeNonSerClass info;

    ...
}
```

Control de Versiones

- ¿Qué pasa si serializamos un objeto, pero mientras tanto la clase sabia un poco (por ejemplo, tiene nuevo atributo) ?
- Cada clase tiene un identificador único (calculado automáticamente en base a varios aspectos como sus atributos, etc) . Si el identificador de la clase serializada no es el mismo que el de la clase usada para la deserialización se lanza una excepción : java.io.InValidClassException
- Para solucionar esto, podemos asignar una versión se utilizará en lugar del identificador de la clase …

```
public class Person implements Serializable {
    private static final long serialVersionUID = 1L;
    …
}
```