Boletín 1. Persistencia de objetos

BDSW Tema 1 –Persistencia 22 de septiembre de 2014

Índice

Índice de tablas

Índice de listados

1 Introducción

La serialización es uno de los mecanismos de Java para dotar de persistencia a las instancias de las clases. Los objetivos de este ejercicio son:

- Utilizar las técnicas de serialización para escribir y leer objetos Java.
- Comprender las ventajas de este mecanismo.
- Descubrir sus limitaciones y carencias y la necesidad de utilizar otros mecanismos de persistencia en las aplicaciones empresariales.

2 Creación del proyecto Eclipse

Para crear nuestro primer proyecto Eclipse abrimos el menú **File** y seleccionamos **New** → **Java Project** . En el panel creamos el proyecto `tasi`. Seleccionamos con el botón derecho el icono del proyecto y en el menú contextual seleccionamos **New** → **Package** y creamos los paquetes `sssi.tasi.capitulo1.clases` y `sssi.tasi.capitulo1.serial`.

2.1 La clase `Empleado`

Dentro del paquete `clases` crearemos la clase *"serializable"* `Empleado` con el código mostrado en el listado 1.

Listado 1: Código de la clase `Empleado.java`

```java
package sssi.tasi.capitulo1.clases;

import java.io.Serializable;
import java.util.Date;

public class Empleado implements Serializable {
  private static final long serialVersionUID = 1L;

  private String idEmpleado;
  private String nombre;
  private String apellidos;
  private String departamento;
  private Date fechaContrato;
  private String puesto;
  private int nivelEducacion;
  private double sueldo;
  private double complemento;

  public Empleado() {
    this.idEmpleado = "";
    this.nombre = "";
    this.apellidos = "";
    this.departamento = "";
    this.fechaContrato = new Date();
    this.puesto = "";
    this.nivelEducacion = 0;
    this.sueldo = 0.0;
    this.complemento = 0.0;
  }

  public Empleado(String idEmpleado, String nombre, String
  apellidos, String departamento, Date fechaContrato, String puesto,
  int nivelEducacion, double sueldo, double complemento) {
    this.idEmpleado = idEmpleado;
    this.nombre = nombre;
    this.apellidos = apellidos;
    this.departamento = departamento;
    this.fechaContrato = fechaContrato;
    this.puesto = puesto;
    this.nivelEducacion = nivelEducacion;
    this.sueldo = sueldo;
    this.complemento = complemento;
  }

  public Empleado(String idEmpleado, String nombre, String apellidos) {
    this.idEmpleado = idEmpleado;
    this.nombre = nombre;
    this.apellidos = apellidos;
    this.departamento = "";
    this.fechaContrato = new Date();
    this.puesto = "";
    this.nivelEducacion = 0;
    this.sueldo = 0.0;
    this.complemento = 0.0;
  }

  // getters
  public String getIdEmpleado() {
    return idEmpleado;
  }
```

```
61    public String getNombre() {
62      return nombre;
63    }
64    public String getApellidos() {
65      return apellidos;
66    }
67    public String getDepartamento() {
68      return departamento;
69    }
70    public Date getFechaContrato() {
71      return fechaContrato;
72    }
73    public String getPuesto() {
74      return puesto;
75    }
76    public int getNivelEducacion() {
77      return nivelEducacion;
78    }
79    public double getSueldo() {
80      return sueldo;
81    }
82    public double getComplemento() {
83      return complemento;
84    }
85
86    // setters
87    public void setIdEmpleado(String idEmpleado) {
88      this.idEmpleado = idEmpleado;
89    }
90    public void setNombre(String nombre) {
91      this.nombre = nombre;
92    }
93    public void setApellidos(String apellidos) {
94      this.apellidos = apellidos;
95    }
96    public void setDepartamento(String departamento) {
97      this.departamento = departamento;
98    }
99    public void setFechaContrato(Date fechaContrato) {
100     this.fechaContrato = fechaContrato;
101   }
102   public void setPuesto(String puesto) {
103     this.puesto = puesto;
104   }
105   public void setNivelEducacion(int nivelEducacion) {
106     this.nivelEducacion = nivelEducacion;
107   }
108   public void setSueldo(double sueldo) {
109     this.sueldo = sueldo;
110   }
111   public void setComplemento(double complemento) {
112     this.complemento = complemento;
113   }
114
115   // Otras funciones de utilidad
116   public String toString() {
117     return "[" + idEmpleado + "] " + nombre + " " + apellidos;
118   }
119 }
```

2.2 La clase `SerialWriter`

Dentro del paquete `serial` creamos ahora la clase `SerialWriter` que creará un objeto del tipo empleado y lo escribirá en un fichero. El código de la clase se muestran en el listado 2

Listado 2: Código de la clase `SerialWriter.java`

```
1  package sssi.tasi.capitulo1.serial;
2  import java.io.*;
3  import java.text.SimpleDateFormat;
4  import java.util.Date;
5  import sssi.tasi.capitulo1.clases.Empleado;
6
7  public class SerialWriter {
8    public static void main(String[] args) {
9      FileOutputStream fout = null;
10     ObjectOutputStream oout = null;
11
12     try {
13       fout = new FileOutputStream("empleado.ser");
14       oout = new ObjectOutputStream(fout);
15
16       SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd");
17
18       Empleado empleado = new Empleado("000003", "PABLO", "IVARS SORIANO"
             );
19       empleado.setDepartamento("004");
20       Date fc =  formatter.parse("2006-07-06");
21       empleado.setFechaContrato(fc);
22       empleado.setPuesto("MGR");
23       empleado.setNivelEducacion(7);
24       empleado.setSueldo(20396.00);
25       empleado.setComplemento(11109.00);
26
27       oout.writeObject(empleado);
28
29     }
30     catch (Exception e) {
31       e.printStackTrace();
32     }
33     finally {
34       try { if (oout != null) oout.close(); }
35       catch (IOException e) {}
36       try { if (fout != null) fout.close(); }
37       catch (IOException e) {}
38     }
39   }
40 }
```

2.3 Ejecución de la clase `SerialWriter`

Para ejecutar la aplicación `SerialWriter` seleccionamos el nombre se la clase con el botón derecho y en el menú contextual seleccionamos **Run As** → **Java Application** . Si la ejecución es correcta, no deberemos obtener ninguna salida en la consola de Eclipse, pero tras realizar un **Refresh** de los ficheros del proyecto deberemos encontrar el fichero `empleado.ser` que contiene el objeto serializado.

2.4 La clase `SerialReader`

Ahora vamos a escribir una clase para recuperar el estado del objeto ("leer") `Empleado`. Para ello creamos la clase `SerialReader` dentro del paquete `serial`. El código de esta clase se muestra en el listado 3.

Listado 3: Código de la clase `SerialReader.java`

```
1  package sssi.tasi.capitulo1.serial;
2
3  import java.text.SimpleDateFormat;
4  import java.io.*;
5  import sssi.tasi.capitulo1.clases.Empleado;
6
7  public class SerialReader {
8    public static void main(String[] args) {
9      FileInputStream fin = null;
10     ObjectInputStream oin = null;
11
12     try {
13       fin = new FileInputStream("empleado.ser");
14       oin = new ObjectInputStream(fin);
15
16       SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd");
17
18       Empleado empleadoGuardado = (Empleado)oin.readObject();
19       System.out.println(empleadoGuardado);
20       System.out.println(" Departamento: " +
21         empleadoGuardado.getDepartamento());
22       System.out.println("       Puesto: " +
23         empleadoGuardado.getPuesto() + ", Antiguedad: " +
24         formatter.format(empleadoGuardado.getFechaContrato()));
25       System.out.println("  Nivel Educ.: " +
26         empleadoGuardado.getNivelEducacion());
27       double total = empleadoGuardado.getSueldo() +
28         empleadoGuardado.getComplemento();
29       System.out.println("       Sueldo: " +
30         empleadoGuardado.getSueldo() + ", Complemento: " +
31         empleadoGuardado.getComplemento() + ", Total: " + total);
32     }
33     catch (Exception e) {
34       e.printStackTrace();
35     }
36     finally {
37       try { if (oin != null) oin.close(); }
38       catch (IOException e) {}
39       try { if (fin != null) fin.close(); }
40       catch (IOException e) {}
41     }
42   }
43 }
```

2.5 Ejecución de la clase `SerialReader`

Para ejecutar la aplicación `SerialReader` seleccionamos el nombre se la clase con el botón derecho y en el menú contextual seleccionamos **Run As** → **Java Application** . Si la ejecución es correcta, obtendremos en la consola de Eclipse los datos del empleado.

3 Complicando el modelo

Imaginemos que nuestra clase `Empleado` forma parte de una aplicación que estamos desarrollando para la gestión de una empresa. Los empleados están asociados a un determinado departamento dentro de la organización. Para mantener la información de los departamentos, diseñamos y creamos la clase `sssi.tasi.capitulo1.clases.Departamento` como se muestra en el listado 4.

Listado 4: Código de la clase `Departamento.java`

```java
package sssi.tasi.capitulo1.clases;

import java.io.Serializable;

public class Departamento implements Serializable {
  private static final long serialVersionUID = 1L;

  private String idDepartamento;
  private String nombre;
  private String manager;

  public Departamento() {
    super();
    this.idDepartamento = "";
    this.manager = "";
    this.nombre = "";

  }
  public Departamento(String idDepartamento,
  String nombre, String manager) {
    super();
    this.idDepartamento = idDepartamento;
    this.manager = manager;
    this.nombre = nombre;
  }

  public String getIdDepartamento() {
    return idDepartamento;
  }
  public String getNombre() {
    return nombre;
  }
  public String getManager() {
    return manager;
  }
  public void setIdDepartamento(String idDepartamento) {
    this.idDepartamento = idDepartamento;
  }
  public void setNombre(String nombre) {
    this.nombre = nombre;
  }
  public void setManager(String manager) {
    this.manager = manager;
  }
}
```

3.1 Actividad 1

Con objeto de mejorar nuestro modelo vamos a realizar las siguientes modificaciones:

- Cree la clase `EmpleadoDao.java` que debe contar con los métodos:
  - `void save(Empleado e, String fileName).`
  - `Empleado read(String fileName).`
- Cree la clase `DepartamentoDao.java` que debe contar con los métodos:
  - `void save(Departamento d, String fileName).`
  - `Departamento read(String fileName).`

- Modifique la clase `SerialWriter` para que cree y almacene el empleado y el departamento en el que trabaja ("SOPORTE DE SOFTWARE", código "E21"). Debe utilizar las clases `EmpleadoDao` y `DepartamentoDao` para esta tarea.
- Modifique la clase `SerialReader` para que recupere los datos del empleado y del departamento y los muestre por la consola. También deberá hacer uso de las clases "`Dao`" anteriores.

3.2 Actividad 2

Nuestro sistema de información ha resultado un éxito y ha propiciado que la empresa se haya posicionado entre las primeras de su sector. Lamentablemente, esto ha obligado a qla contratación de más personal y a la creación de nuevos departamentos, por lo que será necesario adaptar y ampliar el sistema informático a las nuevas necesidades. Realice las siguientes modificaciones:

- Modifique la clase `EmpleadoDao.java` para que cuente con los métodos:
  - `void saveAll(Collection<Empleado>e, String fileName).`
  - `Collection<Empleado>readAll(String fileName).`
- Modifique la clase `DepartamentoDao.java` para que cuente con los métodos:
  - `void saveAll(Collection<Departamento>d, String fileName).`
  - `Collection<Departamento>readAll(String fileName).`
- Construya la clase `MultiSerialWriter` que se encarga del almacenamiento de una colección de empleados. Los recién incorporados a la empresa se muestran en la tabla 1 y los nuevos departamentos en la tabla 2.
- Construya la clase `MultiSerialReader` que lea y muestre por la consola los datos de empleados y departamentos almacenados.

Id	Nombre	Apellidos	Dept.	Antiguedad	Puesto	Edu.	Sueldo	Compl.
000003	PABLO	IVARS SORIANO	004	2006-07-06	MGR	7	20396.00	11109.00
000011	ANA ISABEL	GANDIA BARROSO	004	2004-08-25	ANL	9	28702.00	10875.00
000013	JUAN	ANGULLO GARCIA	003	2005-06-19	DES	9	20139.00	8543.00
000020	LUCIA	CASCALES ABAD	004	2005-11-18	SLS	9	27269.00	6990.00
000022	FERNANDO A.	LLUNA IRANZO	002	2006-01-04	PRES	9	45532.00	1087.00
000033	RAFAEL	BOLUDA ALVAREZ	002	2004-04-02	SLS	5	13725.00	1031.00
000038	ZULEMA	GARCIA FONFRIA	004	2006-04-06	SLS	10	42871.00	2817.00
000042	BLANCA	RIBERA HERRERO	005	2004-03-11	SLS	6	36293.00	10848.00
000049	MARIA	ALMENAR BALLESTER	003	2005-03-18	DES	12	29951.00	2315.00
000054	VANESA	SURIA NAVAS	003	2004-06-10	SLS	9	32331.00	3689.00
000055	CARLOS	ARANDIGA SANCHEZ	004	2007-11-20	FLD	9	28599.00	7325.00
000061	AARON	PEREZ SANCHEZ	004	2005-07-03	SLS	10	17496.00	8575.00

Tabla 1: Relación de nuevos empleados de la empresa.

Id	Nombre	Manager
001	DIRECCION	000020
002	PLANIFICACION	000011
003	CENTRO DE INFORMACION	000013
004	SISTEMAS DE PRODUCCION	000003
005	SOPORTE DE SOFTWARE	000038

Tabla 2: Relación de departaemntos de laempresa.

3.3 Actividad 3

Con objeto de extraer información úitl de nuestro sistema de información, se requiere que realice las siguientes operaciones (implemente en una clase individual cada una de las operaciones descritas:

- Obtenga la lista de empleados clasificada por departamentos.

- Elabore un procedimiento para comprobar la integridad de los datos. Concretamente deberá garantizar que:
  - Existe el departamento en el que trabaja el empleado.
  - Todos los departamentos tienen un *manager* y que además está en la lista de empleados.