1 Ficheros de configuración

En la API de Java se incluyen librerías para trabajar con los ficheros de configuración. Puesto que todos siguen un mismo patrón, es la librería la que se encarga de acceder al fichero a bajo nivel y el programador sólo tiene que indicar a que propiedad quiere acceder o que propiedad quiere modificar, sin tener que añadir nada de código para leer o escribir el fichero tal y como hemos visto en el punto anterior.

Un ejemplo de fichero de configuración de esta librería de java sería el fichero que sigue:

```
<file properties configuracion.props>
# Fichero de configuracion
# Thu Nov 14 10:49:39 CET 2013

user=usuario
password=micontrasena
server=localhost
port=3306
```

Escribir ficheros de configuración

Así, si queremos generar, desde Java, un fichero de configuración como el anterior:

```
...
Properties configuracion = new Properties();
configuracion.setProperty("user", miUsuario);
configuracion.setProperty("password", miContrasena);
configuracion.setProperty("server", elServidor);
configuracion.setProperty("port", elPuerto);
try {
  configuracion.store(new FileOutputStream("configuracion.props"),
                                           "Fichero de configuracion");
} catch (FileNotFoundException fnfe ) {
  fnfe.printStackTrace();
} catch (IOException ioe) {
  ioe.printStackTrace();
}
...
```

Leer ficheros de configuración

A la hora de leerlo, en vez de tener que recorrer todo el fichero como suele ocurrir con los ficheros de texto, simplemente tendremos que cargarlo e indicar de qué propiedad queremos obtener su

valor (`getProperty(String)`).

```
...
Properties configuracion = new Properties();
try {
  configuracion.load(new FileInputStream("configuracion.props"));
  usuario = configuracion.getProperty("user");
  password = configuracion.getProperty("password");
  servidor = configuracion.getProperty("server");
  puerto = Integer.valueOf(configuration.getProperty("port"));
} catch (FileNotFoundException fnfe ) {
  fnfe.printStackTrace();
} catch (IOException ioe) {
  ioe.printStackTrace();
}
...
```

Para ambos casos, escribir y leer este tipo de ficheros, hay que tener en cuenta que, al tratarse de ficheros de texto, toda la información se almacena como si de un `String` se tratara. Por tanto, todos aquellos tipos `Date` , `boolean` o incluso cualquier tipo numérico serán almacenados en formato texto. Así, habrá que tener en cuenta las siguientes consideraciones:

- Para el caso de las fechas, deberán ser convertidas a texto cando se quieran escribir y nuevamente reconvertidas a `Date` cuando se lea el fichero y queramos trabajar con ellas
- Para el caso de los tipos `boolean` , podemos usar el método `String.valueOf(boolean)` para pasarlos a `String` cuando queramos escribirlos. En caso de que queramos leer el fichero y pasar el valor a tipo `boolean` podremos usar el método `Boolean.parseBoolean(String)`
- Para el caso de los tipos numéricos (`integer` , `float` , `double`) es muy sencillo ya que Java los convertirá a `String` cuando sea necesario al escribir el fichero. En el caso de que queramos leerlo y convertirlos a su tipo concreto, podremos usar los métodos `Integer.parseInt(String)` , `Float.parseFloat(String)` y `Double.parseDouble()` , según proceda