U1
Accés a fitxers

Accés a dades DAM

Continguts

1. Propietats del sistema
2. Tipus de fitxers
3. Fitxers de text pla
4. Fitxers de configuració
5. Fitxers XML
6. Fitxers binaris. Serialització.

1. Propietats del sistema

Per poder accedir als fitxers, és convenient conèixer les propietats del sistema. Per això, podem utilitzar System Properties de Java, amb les quals podem accedir a les propietats de configuració del sistema. Algunes es poden interessen conèixer en aquesta unitat:

- **file.separator**: Obté el caràcter que utilitza el S.O. per a la separació de les rutes (\ o /).
- **user.home**: Obté la ruta de la carpeta personal de l'usuari
- **user.dir**: Obté la ruta on es troba actualment l'usuari
- **line.separator**: Obté el caràcter que separa les línies d'un fitxer de text

Totes aquestes propietats depenen del S.O. que estiguem utilitzant i/o de l'usuari utilitzat.

2. Tipus de fitxers

Fitxers de text: Inclouen exclusivament caràcters de text. Els podeu llegir des d'un editor de text.

Extensió	Descripció
.txt	Fitxer de text pla
.xml	Fitxer XML
.json	Fitxer d'intercanvi d'informació
.props	Fitxer de propietats
.conf	Fitxer de configuració
.sql	Script SQL
.srt	Fitxer de subtítol

2. Tipus de fitxers

Fitxers binaris: [redacted]

Extensió	Descripció
.pdf	Fitxer PDF
.jpg	Fitxer d'imatge
.doc, .docx	Fitxer de Microsoft Word *
.avi	Fitxer de vídeo
.ppt, .pptx	Fitxer de Power Point

També ens podem trobar amb fitxers binaris amb extensió **.bin** i **.dat**. Aquests són fitxers que es llegeixen/escriuen de manera específica només definida per una aplicació concreta.

* .doc, .docx: tot i que poden contenir només text, en emmagatzemar-se el fitxer el processador inclou informació binària.

3. Fitxers de text pla

Els **fitxers de text** són aquells que **únicament contenen text**, per la qual cosa poden ser editats directament amb qualsevol **editor de text pla** (Bloc de Notes, notepad++, . . .).

Són aquells que normalment s'emmagatzemen amb l'extensió .txt però també podem incloure els **scripts SQL** (.sql), fitxers de **codi Java** (.java), fitxers de **configuració** (.ini, .props, .conf, . . .)...

També s'inclouen a la categoria de fitxers de text pla els que a més inclouen informació addicional (sempre en forma de text) que permeten interpretar les dades del fitxer d'una manera o altra, afegint-hi més informació. Aquests formats són **HTML**, **XML**, **JSON**,...

3. Fitxers de text pla - Escriptura

```
FileWriter fitxer = null;
PrintWriter escriptor = null;
try {
  fitxer = new FileWriter("fitxer.txt");
  escriptor = new PrintWriter(fitxer) ;
  escriptor.println("Nova línia del fitxer.");
} catch (IOException ioe) {
  ioe.printStackTrace() ;
} finally {
if (fitxer != null)
  try {
   fitxer.close();
  } catch (IOException ioe) { . . . }
}
```

3. Fitxers de text pla - Lectura

```
File fitxer = null;
FileReader lector = null;
BufferedReader buffer = null;

try {
  fitxer = new File("fitxer.txt");
  lector = (new FileReader(fitxer));
  buffer = new BufferedReader(lector);
  String linia = null;
  while ((linia = buffer.readLine()) != null)
    System.out.println(linia);
} catch (FileNotFoundException fnfe) {
  fnfe.printStackTrace();
} catch (IOException ioe) {
  ioe.printStackTrace();
} finally {
  if (buffer != null)
  try {
    buffer.close();
  } catch (IOException ioe) { . . . }
}
```

4. Fitxers de configuració

A Java s'inclouen llibreries per treballar amb els fitxers de configuració. Aquesta llibreria és la que s'encarrega d'accedir al fitxer a baix nivell i el programador només ha d'indicar a quina propietat vol accedir o quina propietat vol modificar, sense haver d'afegir res de codi per llegir o escriure el fitxer tal com hem vist al punt anterior.

A continuació es mostra un exemple de fitxer de configuració d'aquesta llibreria de java:

config.props

```
# Config file
# Thu Nov 14 10:49:39 CET 2013
user=user
password=mypassword
server=localhost
port=3306
```

4. Fitxers de configuració - Escriptura

Generar desde Java, un fitxer de configuració com l'anterior:

```
. . .
Properties configuracio = new Properties();
configuracio.setProperty("user, “user_”);
configuracio.setProperty("password", “password_”);
configuracio.setProperty("server", “server_”);
configuracio.setProperty("port", “port_”);
try {
  configuracio.store(new FileOutputStream("configuracio.props"),
                                          "fitxer de configuracio");
} catch (IOException ioe) {
  ioe.printStackTrace();
}
. . .
```

4. Fitxers de configuració - Lectura

A l'hora de llegir el fitxer de configuració, en comptes d'haver de recórrer tot el fitxer com sol passar amb els fitxers de text, simplement haurem de carregar-lo i indicar de quina propietat volem obtenir-ne el valor amb getProperty(String)).

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

4. Fitxers de configuració - Lectura

Com **tota la informació s'emmatzema com a String**, tots aquells tipus Date, boolean o fins i tot qualsevol tipus numèric seran emmagatzemats en format text. Així, caldrà tenir en compte les consideracions següents:

- Per al cas de les **dates**, hauran de ser convertides a text quan es vulguin escriure i novament reconvertides a *Date* quan es llegeixi el fitxer i vulguem treballar-hi
- Per al cas dels tipus **boolean**, podem fer servir el mètode S*tring.valueOf(boolean)* per passar-los a String quan vulguem escriure'ls. En cas que vulguem llegir el fitxer i passar el valor a tipus boolean podrem fer servir el mètode *Boolean.parseBoolean(String)*
- Per al cas dels tipus **numèrics** (integer, float, double) és molt senzill ja que Java els convertirà a String quan sigui necessari en escriure el fitxer. En cas que vulguem llegir-lo i convertir-los al seu tipus concret, podrem utilitzar els mètodes *Integer.parseInt(String)*, *Float.parseFloat(String)* i *Double.parseDouble()*, segons escaigui.

5. Fitxers XML

Els **fitxers XML** permeten l'**intercanvi d'informació entre aplicacions utilitzant un fitxer de text pla** al qual se li poden afegir etiquetes per donar significat a cadascun dels valors que s'emmagatzemen.

5. Fitxers XML

El següent fitxer XML podria ser el resultat de bolcar una Base de Dades sobre productes d'una companyia, de manera que aquesta informació ara es podria llegir des d'una altra aplicació i incorporar-la. Es pot veure com a part de les dades d'aquests productes (els seus noms, preus, . . .) apareix una altra informació en forma d'etiquetes (entre els caràcters < i >) que permet donar significat a cada dada emmagatzemada al fitxer. Així, podem saber de què parlem ia què correspon cada valor.

Tal com passava amb els fitxers de configuració, **tota la informació s'ha de passar a text per poder crear el fitxer i s'ha de reconvertir al seu tipus original quan es carregui de nou**. Les mateixes consideracions que hem tingut en compte abans per a la conversió de tipus de dades ens serviran ara.

```
<?xml version="1.0"
encoding="UTF-8" standalone="no">
<xml>
<productos>
  <producto>
    <nombre>Cereales</nombre>
    <precio>3.45</precio>
  </producto>
  <producto>
    <nombre>Colacao</nombre>
    <precio>1.45</precio>
  </producto>
  <producto>
    <nombre>Agua mineral</nombre>
    <precio>1.00</precio>
  </producto>
</productos>
</xml>
```

5. Fitxers XML

La classe Java que defineix cadascun dels objectes que es representen pel fitxer XML de l'exemple, és la següent:

```java
public class Producto {
  private String nombre;
  private float precio;

  public Producto(String nombre, float precio) {
    this.nombre = nombre;
    this.precio = precio;
  }

  public String getNombre () { return nombre; }
  public void setNombre (String nombre) { this.nombre = nombre; }

  public float getPrecio () { return precio; }
  public void setPrecio (float precio) { this.precio = precio; }
}
```

```xml
<?xml version="1.0"
encoding="UTF-8" standalone="no">
<xml>
<productos>
  <producto>
    <nombre>Cereales</nombre>
    <precio>3.45</precio>
  </producto>
  <producto>
    <nombre>Colacao</nombre>
    <precio>1.45</precio>
  </producto>
  <producto>
    <nombre>Agua mineral</nombre>
    <precio>1.00</precio>
  </producto>
</productos>
</xml>
```

5. Fitxers XML - Lectura

Si volem llegir un fitxer XML i carregar-lo com una col·lecció Java d'objectes Producte, l'exemple següent mostra un breu exemple sobre com accedir a la informació del fitxer XML:

```
. . .
NodeList productos = documento.getElementyByTagName("producto");
for (int i = 0; i < productos.getLength(); i++) {
  Node producto = productos . item ( i ) ;
  Element elemento = ( Element ) producto ;
  System.out.println(elemento.getElementsByTagName("nombre").item(0)
                     .getChildNodes().item(0).getNodeValue());
  System.out.println(elemento.getElementsByTagName("precio").item(0)
                     .gethildNodes().item(0).getNodeValue());
}
. . .
```

5. Fitxers XML - Escriptura

Si ara volem generar el fitxer XML amb tota la informació d'una col·lecció d'objectes Producte, podem utilitzar el codi següent:

```
. . .
documento = dom.createDocument(null, "xml", null);
Element raiz = document.createElement("productos");
documento.getDocumentElement().appendChild(raiz);
Element nodoProducto = null , nodoDatos = null ;
Text texto = null;

for (Producto producto : listaProductos) {
  nodoProducto = documento.createElement("producto");
  raiz.appendChild(nodoProducto);
  nodoDatos = documento.createElement("nombre");
  nodoProducto.appendChild(nodoDatos);
  texto = documento.createTextNode(producto.getNombre());
  nodoDatos.appendChild(texto);
  nodoDatos = documento.createElement("precio");
  nodoProducto.appendChild(nodoDatos);
  texto = documento.createTextNode(producto.getPrecio());
  nodoDatos.appendChild(texto);
}
. . .
```

6. Fitxers binaris

Per realitzar la lectura i escriptura de fitxers binaris utilitzarem un procés anomenat serialització.

Serialitzar és el procés pel qual un objecte en memòria passa a transformar-se en una estructura que es pugui emmagatzemar en un fitxer (persistència).

- Al procés contrari l'anomenarem deserialitzar.
- **Cada objecte es serialitza a un fitxer**
- **Si volem emmagatzemar tots els objectes d'una aplicació, la idea més convenient és tenir tots aquests en alguna estructura complexa** (i per comoditat dinàmica) de manera que sigui aquesta estructura la que serialitzem o deserialitzem per guardar o carregar els objectes d'una aplicació.
- Exemples d'estructures que podem serialitzar o deserialitzar: ArrayList, Set o HashMap són classes molt utilitzades per treballar així.

6. Fitxers binaris - Serialització

```
. . .
ObjectOutputStream serializador = null;
try {
  serializador = new ObjectOutputStream(new FileOutputStream("fitxer.dat"));
  serializador.writeObject(listaProductos);
} catch (IOException ioe) {
  . . .
} finally {
  if (serializador != null)
    try {
      serializador.close();
    } catch (IOException ioe) {
      ioe.printStackTrace();
    }
  }
. . .
```

6. Fitxers binaris - Deserialització

```
. . .
List<Producto> listaProductos = null;
ObjectInputStream deserializador = null;
try {
  deserializador = new ObjectInputStream(new
                                    FileInputStream("fitxer.dat"));
  listaProductos = (ArrayList<Producto>)deserializador.readObject();
} catch (FileNotFoundException fnfe ) {
  fnfe.printStackTrace();
} catch (ClassNotFoundException cnfe ) {
  cnfe.printStackTrace();
} catch (IOException ioe) {
  ioe.printStackTrace();
} finally {
  if (deserializador != null)
  try {
    deserializador.close();
  } catch (IOException ioe) {
    ioe.printStackTrace();
  }
}
. . .
```