Volver

U7.3 - Gestión de Archivos con la clase File

Volver

Introducción a la clase File

Volver

¿Qué es la clase File?

- La clase `File` permite manipular rutas del sistema de archivos en Java y Kotlin.
- No solo sirve para archivos: también para carpetas o rutas que no existen aún.
- Es necesario importar `java.io.File` para usarla.
- Hay que instanciar un objeto `File` con la ruta deseada.
- La ruta puede ser a un archivo o a una carpeta, existente o no.
- El objeto representa una única ruta; para varias rutas, varios objetos.

Volver

Ejemplo de uso básico

```
import java.io.File

val ruta = "data.txt"
val f = File(ruta) // Instancia un objeto File para la ruta especificada
```

- Se puede usar tanto para archivos como para carpetas.
- El objeto File aún no accede al sistema de archivos hasta usar sus métodos.

Volver

Rutas Absolutas y Relativas

Volver

Rutas absolutas

- Una ruta absoluta parte del directorio raíz del sistema de archivos.
- En Windows: comienzan con la unidad (ej. `C:\Fotos\Viajes`).
- En Unix/Linux: comienzan con `/` (ej. `/Fotos/Viajes`).
- Son útiles para identificar exactamente un archivo o carpeta.
- Requieren que la estructura de carpetas exista igual en todos los equipos.

Volver

Rutas relativas

- No incluyen el directorio raíz, parten del directorio de trabajo de la aplicación.
- El directorio de trabajo suele ser donde se lanza el programa.
- Facilitan la portabilidad del software entre distintos sistemas.
- Ejemplo: `Unidad11/apartado1/Actividades.txt` es relativo.
- El mismo código puede funcionar en diferentes equipos si se mantiene la estructura relativa.

Volver

Métodos principales de la clase File

Volver

Métodos para obtener información de la ruta

- `getParent()`: devuelve la ruta del directorio padre.
- `getName()`: devuelve el nombre del archivo o carpeta.
- `getAbsolutePath()`: devuelve la ruta absoluta completa.
- Permiten conocer y mostrar información sobre la ruta gestionada.

Volver

Comprobaciones de estado

- `exists()`: indica si la ruta existe en el sistema de archivos.
- `isFile()`: indica si la ruta existe y es un archivo.
- `isDirectory()`: indica si la ruta existe y es una carpeta.
- Útiles para validar la existencia y tipo antes de operar.

Volver

Propiedades de archivos

- `length()`: devuelve el tamaño del archivo en bytes.
- `lastModified()`: devuelve la fecha de última modificación en milisegundos desde 1970.
- Permiten consultar atributos importantes de archivos.

Volver

Gestión de Archivos y Carpetas

Volver

Operaciones básicas

- `mkdir()`: crea una carpeta en la ruta especificada.
- `delete()`: borra el archivo o carpeta (solo si la carpeta está vacía).
- `renameTo(destino)`: mueve o renombra un archivo o carpeta.
- Devuelven `true` si la operación fue exitosa, `false` en caso contrario.

Volver

Listado de archivos en una carpeta

- `listFiles()`: devuelve un array de objetos File con el contenido de la carpeta.
- Solo funciona si la ruta es una carpeta existente.
- El orden de los elementos es aleatorio.

Volver

Creación de Archivos en Kotlin

Volver

Crear archivos de distintas formas

- `createNewFile()`: crea un archivo vacío si no existe, devuelve `true` si lo crea.
- `writeText(text)`: crea o sobrescribe el archivo y escribe texto.
- `writeBytes(bytes)`: crea o sobrescribe el archivo y escribe un array de bytes.
- Si el archivo ya existe, `writeText` y `writeBytes` lo sobrescriben.

Note: Presenta los métodos más comunes para crear archivos en Kotlin, destacando el comportamiento ante archivos existentes.

Usando `createNewFile()`

Volver

Ejemplo: Crear archivo con createNewFile

```
import java.io.File

val file = File("data.txt")
val creado = file.createNewFile()
if (creado) {
    println("Archivo creado.")
} else {
    println("El archivo ya existe.")
}
```

- Crea el archivo solo si no existe, evitando sobrescribir datos.

Note: Ejemplo práctico de uso de createNewFile, mostrando cómo evitar sobrescribir archivos existentes.

Usando `writeText()`

- Crea archivo y escribe texto (UTF-8 por defecto).
- Sobrescribe si existe, perdiendo datos previos.

Volver

Ejemplo: Crear archivo con writeText

```
import java.io.File

val file = File("data.txt")
file.writeText("Contenido de ejemplo") // Sobrescribe si ya existe
```

- Escribe texto en el archivo, sobrescribiendo el contenido anterior.

Volver

Usando `writeBytes()`

- Crea archivo y escribe array de bytes.
- Sobrescribe si existe, perdiendo datos previos.
- Similar a `writeText()`, pero con datos binarios.

Volver

Ejemplo: Crear archivo con writeBytes

```
import java.io.File

val file = File("data.txt")
file.writeBytes(ByteArray(0)) // Escribe un array de bytes vacío
```

- Permite escribir datos binarios en el archivo.

Volver