



## Gérer le « système de fichiers » avec la classe File

---

Xavier HER

1



## Responsabilité de la classe File

- Représentation abstraite d'un fichier ou d'un répertoire permettant de tester l'existence, l'état, de le renommer, le détruire, ...
  - Abstraction du fichier
- Package java.io
- Exceptions contrôlées java.io.IOException

XH

2



## La classe File

File	
<ul style="list-style-type: none"> <li>+ separatorChar : char</li> <li>+ pathSeparatorChar : char</li> <li>+ separator : String</li> <li>+ pathSeparator : String</li> </ul>	
<ul style="list-style-type: none"> <li>+ File(argC : String)</li> <li>+ File(argC : String arg' : String)</li> <li>+ File(argC : File arg' : String)</li> <li>+ getName() : String</li> <li>+ getParent() : String</li> <li>+ getParentFile() : File</li> <li>+ getPath() : String</li> <li>+ isAbsolute() : boolean</li> <li>+ getAbsolutePath() : String</li> <li>+ getAbsoluteFile() : File</li> <li>+ getCanonicalPath() : String</li> <li>+ getCanonicalFile() : File</li> <li>+ toURL() : URL</li> <li>+ canRead() : boolean</li> <li>+ canWrite() : boolean</li> <li>+ exists() : boolean</li> <li>+ isDirectory() : boolean</li> <li>+ isFile() : boolean</li> <li>+ isHidden() : boolean</li> <li>+ lastModified() : long</li> <li>+ length() : long</li> <li>..</li> </ul>	<ul style="list-style-type: none"> <li>..</li> <li>+ createNewFile() : boolean</li> <li>+ delete() : boolean</li> <li>+ deleteOnExit() : void</li> <li>+ list() : String[]</li> <li>+ list(argC : FilenameFilter) : String[]</li> <li>+ listFiles() : File[]</li> <li>+ listFiles(argC : FilenameFilter) : File[]</li> <li>+ listFiles(argC : FileFilter) : File[]</li> <li>+ mkdir() : boolean</li> <li>+ mkdirs() : boolean</li> <li>+ renameTo(argC : File) : boolean</li> <li>+ setLastModified(argC : long) : boolean</li> <li>+ setReadOnly() : boolean</li> <li>+ listRoots() : File[]</li> <li>+ createTempFile(argC : String arg' : String arg'' : String) : File</li> <li>+ createTempFile(argC : String arg' : String) : File</li> <li>+ compareTo(argC : File) : int</li> <li>+ compareTo(argC : Object) : int</li> <li>+ equals(argC : Object) : boolean</li> <li>+ hashCode() : int</li> <li>+ toString() : String</li> </ul>

XH ■ API J2SE 1.3

3



## Exemple avec File (1)

### ■ Vérifier l'existence d'un fichier

```
import java.io.*;
import javax.swing.JOptionPane;
public class VerifierExistenceFichier {
    public static void main (String args[]) {
        try{
            File fic = new File("C:/octets.dat");
            if (fic.exists()){
                String reponse = JOptionPane.showInputDialog(
                    "Voulez-vous détruire le contenu existant (oui ou non)?");

                if((reponse.toUpperCase()).equals("NON")){
                    System.out.println("Le fichier est inchangé");
                    System.exit(0);
                }
            }
            FileOutputStream flux = new FileOutputStream( fic );
            flux.close();
        }catch( Exception ex){
            ex.printStackTrace();
        }
    }
}
```

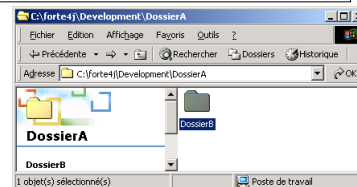
4



## Exemple avec File (2)

### ■ Créer un répertoire

```
import java.io.*;
public class CreerRepertoire{
public static void main (String args[]) throws IOException
{
    File unFile = new File("C:/DossierA/DossierB/");
    unFile.mkdirs();
    if(unFile.exists())
        System.out.println("Le répertoire a été créé");
    else
        System.out.println("Le répertoire n'a pas été créé");
}}
```

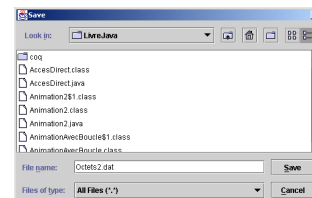


XH

5



## Dialogue de sélection de fichier avec la classe JFileChooser



```
import java.io.*;
import javax.swing.*;
public class CreerFichierFileChooser extends JFrame {
public CreerFichierFileChooser() throws IOException {
    JFileChooser unFileChooser = new JFileChooser();
    unFileChooser.setFileSelectionMode(JFileChooser.FILES_ONLY);
    int resultat = unFileChooser.showSaveDialog(this);
    if (resultat != JFileChooser.CANCEL_OPTION) {
        File leFile = unFileChooser.getSelectedFile();
        if (leFile != null && !(leFile.getName().equals("")) {
            FileOutputStream unFichier = new FileOutputStream(leFile);
            unFichier.close();
        }
    }
}
public static void main(String[] args) throws Exception {
    CreerFichierFileChooser cc = new CreerFichierFileChooser();
    System.exit(0);
}}
```

XH

6




END

THE END

XH

7



## La classe File

- API J2SE 1.3

XH

File (from io)
+ separatorChar : char
+ pathSeparatorChar : char
+ separator : String
+ pathSeparator : String
+ File(arg0 : String)
+ File(arg0 : String, arg1 : String)
+ File(arg0 : File, arg1 : String)
+ getName() : String
+ getParent() : String
+ getParentFile() : File
+ getPath() : String
+ isAbsolute() : boolean
+ getAbsolutePath() : String
+ getAbsoluteFile() : File
+ getCanonicalPath() : String
+ getCanonicalFile() : File
+ toURL() : URL
+ canRead() : boolean
+ canWrite() : boolean
+ exists() : boolean
+ isDirectory() : boolean
+ isFile() : boolean
+ isHidden() : boolean
+ lastModified() : long
+ length() : long
+ createNewFile() : boolean
+ delete() : boolean
+ deleteOnExit() : void
+ list() : String[]
+ list(arg0 : FilenameFilter) : String[]
+ listFiles() : File[]
+ listFiles(arg0 : FilenameFilter) : File[]
+ listFiles(arg0 : FileFilter) : File[]
+ mkdir() : boolean
+ mkdirs() : boolean
+ renameTo(arg0 : File) : boolean
+ setLastModified(arg0 : long) : boolean
+ setReadOnly() : boolean
+ listRoots() : File[]
+ createTempFile(arg0 : String, arg1 : String, arg2 : File) : File
+ createTempFile(arg0 : String, arg1 : String) : File
+ compareTo(arg0 : File) : int
+ compareTo(arg0 : Object) : int
+ equals(arg0 : Object) : boolean
+ hashCode() : int
+ toString() : String