The Java™ Tutorials

Trail: Essential Classes
Lesson: Basic I/O
Section: File I/O (Featuring NIO.2)

The Java Tutorials have been written for JDK 8. Examples and practices described in this page don't take advantage of improvements introduced in later releases.

Other Useful Methods

A few useful methods did not fit elsewhere in this lesson and are covered here. This section covers the following:

Determining MIME Type

To determine the MIME type of a file, you might find the `probeContentType(Path)` method useful. For example:

```
try {
    String type = Files.probeContentType(filename);
    if (type == null) {
        System.err.format("'%s' has an" + " unknown filetype.%n", filename);
    } else if (!type.equals("text/plain") {
        System.err.format("'%s' is not" + " a plain text file.%n", filename);
        continue;
    }
} catch (IOException x) {
    System.err.println(x);
}
```

Note that `probeContentType` returns null if the content type cannot be determined.

The implementation of this method is highly platform specific and is not infallible. The content type is determind by the platform's default file type detector. For example, if the detector determines a file's content type to be `application/x-java` based on the `.class` extension, it might be fooled.

You can provide a custom `FileTypeDetector` if the default is not sufficient for your needs.

The `Email` example uses the `probeContentType` method.

Default File System

To retrieve the default file system, use the `getDefault` method. Typically, this `FileSystems` method (note the plural) is chained to one of the `FileSystem` methods (note the singular), as follows:

```
PathMatcher matcher =
    FileSystems.getDefault().getPathMatcher("glob:*.*");
```

Path String Separator

The path separator for POSIX file systems is the forward slash, `/`, and for Microsoft Windows is the backslash, `\`. Other file systems might use other delimiters. To retrieve the `Path` separator for the default file system, you can use one of the following approaches:

```
String separator = File.separator;
String separator = FileSystems.getDefault().getSeparator();
```

The `getSeparator` method is also used to retrieve the path separator for any available file system.

File System's File Stores

A file system has one or more file stores to hold its files and directories. The *file store* represents the underlying storage device. In UNIX operating systems, each mounted file system is represented by a file store. In Microsoft Windows, each volume is represented by a file store: `C:`, `D:`, and so on.

To retrieve a list of all the file stores for the file system, you can use the `getFileStores` method. This method returns an `Iterable`, which allows you to use the enhanced for statement to iterate over all the root directories.

```
for (FileStore store: FileSystems.getDefault().getFileStores()) {
   ...
}
```

If you want to retrive the file store where a particular file is located, use the `getFileStore` method in the `Files` class, as follows:

```
Path file = ...;
FileStore store= Files.getFileStore(file);
```

The `DiskUsage` example uses the `getFileStores` method.

Previous page: Watching a Directory for Changes
Next page: Legacy File I/O Code