Accessing the File System

Default File System

To retrieve the default file system, use the `getDefault()` method of the `FileSystems` factory class. 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:*.*");
```

A `Path` instance is always bound to a file system. If no file system is provided when a path is created, then the default file system is used.

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 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.

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.

```
FileSystem fileSystem = FileSystems.getDefault();
for (FileStore store: fileSystem.getFileStores()) {
    System.out.println(store.name() + " - " + store.type());
}
```

On a Windows machine, you will get this kind of result.

```
Windows - NTFS
Data - NTFS
Video - NTFS
Transfer - Fat32
```

If you need to access the drive letters, you can use the following code. Remember that some drive letters may be used without the drive been mounted. The following code checks if every drive letters is readable.

```
for (Path directory : fileSystem.getRootDirectories()) {
    boolean readable = Files.isReadable(directory);
    System.out.println("directory = " + directory + " - " + readable);
}
```

Running the previous code on Windows will give a result similar to this one.

```
directory = C:\ - true
directory = D:\ - true
```

```
directory = E:\ - true
directory = F:\ - false
directory = G:\ - false
```

Last update: *January 25, 2023*

Working with Paths	←	**Accessing the File System**	→	Manipulating Files and Directories

Home > Tutorials > The Java I/O API > File System Basics > Accessing the File System