L53

Monday, February 7, 2022 12:22 PM

File class and Path class, File Copying/Moving

Understanding path

Q1. IO stands for input and output. The java.io package has classes which help us read input from sources and write output to destinations.

These sources and destinations can be files, sockets or input and output streams of other processes.

Java makes IO programming very easy with its extensive classes distributed mainly in two packages java.io and java.nio.file.

We will first learn about the classes which deal with files and file systems in the java.nio.file package and later we will visit the classes which deal with IO streams present in the java.io package.

File system is the one responsible for storing and retrieving data from a storage. File systems usually store the data in files and directories in a hierarchical structure (tree structure).

The starting point for such a hierarchy is called a root node.

Linux, Unix and other Unix-like operating systems use / (forward slash) to denote the root node (there is no name, it is called the root or slash).

Microsoft Windows allows for multiple root nodes which are also called as drivers (C:\, D:\ etc).

Data is stored in named entities called files. These files are grouped under other type of named entities called directories or folders.

A directory can have files and other directories which are called subdirectories of the current parent directory.

The **location** of a **file** or a **directory** in a file system is called **path**.

There are two types of paths - **absolute** and **relative**. For example :

Unix and Unix-like	Microsoft Windows
/home/user/abc.txt	C:\home\user\abc.txt

The **above two** are examples of **absolute paths** because they specify the location of a file named **abc.txt** from the root nodes in their respective operation systems.

A **relative path** does not include the root node and can represent **a file name** or **a directory name** or **a portion of an absolute path** without the root node.

For example, relative paths from the above absolute path can be any of the following:

Unix and Unix-like	Microsoft Windows
home/user/abc.txt	\home\user\abc.txt
home/user	home\user
home	home
user	user
user/abc.txt	user\abc.txt
abc.txt	abc.txt

java.nio.file package contains a class called Path which models the above said path of a file or a directory.

Please note that a Path object is only a representation of a location. It does not mean that a file or a directory must exist at that path.

java.nio.file package also contains a utility class called Files which contains static methods for creating, deleting and manipulating files and directories represented by a Path object.

See and retype the below code. Please note that we need to import **java.nio.file** package to work with the above two classes and we need to import **java.io** package for IOException.

```
package q11352;
import java.io.*;
import java.nio.file.*;
public class PathDemo {
    public static void main(String[] args) {
        Path currentWorkingDirPath = Paths.get("");
        System.out.println("currentWorkingDirPath.toAbsolutePath() : " + currentWorkingDirPath.toAbsolutePath());
        Path path1 = Paths.get("project1");
        System.out.println("path1.toAbsolutePath() : " + path1.toAbsolutePath());
        System.out.println("Does path1 exists? : " + Files.exists(path1));
        try {
            if (Files.exists(path1)) {
                Files.delete(path1);
            }
            Files.createDirectory(path1);
        } catch (FileAlreadyExistsException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
        System.out.println("Does path1 exists? after createDirectory call : " + Files.exists(path1));
        System.out.println("Files.isRegularFile(path1) : " + Files.isRegularFile(path1));
        System.out.println("Files.isDirectory(path1) : " + Files.isDirectory(path1));
    }
}
```

File Copying/Moving

Q2. The Files utility class contains many static methods which work on files and directories represented by Path objects.

Files class provides two easy to use methods for copying and moving. Their method signatures are given below:

1. copy(Path source, Path target, CopyOption... options) - copies the contents represented by source path to the (target) destination path.
2. move(Path source, Path target, CopyOption... options) - moves the contents represented by source path to the (target) destination path.

The class named Files also provides a convenient method to write bytes to small files write(Path path, byte[] bytes, OpenOption... option), it is suggested to be used only for small files. We will later learn efficient ways of reading and writing data to large files. It also has a corresponding method for reading all bytes, which should be used only for small files.

The StandardOpenOption.CREATE when passed to the above mentioned Files.write as the OpenOption, the write method creates the file if the file does not exists and will not throw an exception if it already exists. StandardOpenOption.TRUNCATE_EXISTING option will make sure that if the file exists, the old content is deleted.

```java
package q11353;
import java.util.*;
import java.io.*;
import java.nio.file.*;
public class FilesDemoSample {
    public static void main(String[] args) {
        Path currentWorkingDirPath = Paths.get("");
        System.out.println("currentWorkingDirPath.toAbsolutePath() : " + currentWorkingDirPath.toAbsolutePath());
        Path path = Paths.get("FilesDemoSample.txt");
        Path pathCopy = Paths.get("FilesDemoSampleCopy.txt");
        Path pathMoved = Paths.get("FilesDemoSampleMoved.txt");
        System.out.println("Files.exists(path) : " + Files.exists(path));
        System.out.println("Files.exists(pathCopy) : " + Files.exists(pathCopy));
        System.out.println("Files.exists(pathMoved) : " + Files.exists(pathMoved));
        String text = "Current time is : " + new Date();
        byte[] contentArr = text.getBytes();
        try {
            Files.write(path, contentArr, StandardOpenOption.CREATE, StandardOpenOption.TRUNCATE_EXISTING);
            Files.copy(path, pathCopy, StandardCopyOption.REPLACE_EXISTING);
            Files.move(path, pathMoved, StandardCopyOption.REPLACE_EXISTING);
        } catch (IOException e) {
            e.printStackTrace();
        }
        System.out.println("After writing, copying and moving");
        System.out.println("path.toAbsolutePath() : " + path.toAbsolutePath());
        System.out.println("Files.exists(path) : " + Files.exists(path));
        System.out.println("pathCopy.toAbsolutePath() : " + pathCopy.toAbsolutePath());
        System.out.println("Files.exists(pathCopy) : " + Files.exists(pathCopy));
        System.out.println("pathMoved.toAbsolutePath() : " + pathMoved.toAbsolutePath());
        System.out.println("Files.exists(pathMoved) : " + Files.exists(pathMoved));
    }
}
```