

Java Delete Files

Topics : [JAVA](#)

Written on [April 11, 2023](#)

In Java, you can delete files using the File class.

Here's an example:

```
import java.io.File;

public class DeleteFileExample {
    public static void main(String[] args) {
        File file = new File("file.txt");
        if (file.delete()) {
            System.out.println("File deleted successfully.");
        } else {
            System.out.println("Failed to delete the file.");
        }
    }
}
```

In this example, we create a File object for a file named "file.txt". We then call the delete() method of the File object to delete the file. The delete() method returns true if the file was deleted successfully, or false if the file could not be deleted.

If the file is successfully deleted, we print a message to the console indicating that the file was deleted. If the file could not be deleted, we print a message indicating that the deletion failed.

Note that the delete() method can also be used to delete empty directories. If you want to delete a directory and all its contents, you can use the Files class and the walkFileTree() method to traverse the directory tree and delete all the files and directories. Here's an example:

```
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;

public class DeleteDirectoryExample {
    public static void main(String[] args) {
        Path directory = Paths.get("directory");
        try {
```

```

        Files.walkFileTree(directory, new DeleteFileVisitor());
        System.out.println("Directory deleted successfully.");
    } catch (IOException e) {
        e.printStackTrace();
    }
}

import java.io.IOException;
import java.nio.file.*;
import java.nio.file.attribute.BasicFileAttributes;

public class DeleteFileVisitor extends SimpleFileVisitor<Path> {
    @Override
    public FileVisitResult visitFile(Path file, BasicFileAttributes attrs)
throws IOException {
        Files.delete(file);
        return FileVisitResult.CONTINUE;
    }

    @Override
    public FileVisitResult postVisitDirectory(Path dir, IOException exc)
throws IOException {
        Files.delete(dir);
        return FileVisitResult.CONTINUE;
    }
}

```

In this example, we create a Path object for a directory named "directory". We then use the Files class and the walkFileTree() method to traverse the directory tree and delete all the files and directories. We pass an instance of the DeleteFileVisitor class to the walkFileTree() method, which deletes each file and directory as it is encountered.

If the directory is successfully deleted, we print a message to the console indicating that the directory was deleted. If an IOException occurs while deleting the directory, we catch it and print the stack trace.

© Copyright **Aryatechno**. All Rights Reserved. Written tutorials and materials by [Aryatechno](#)