File API

saacsos

File System

Path คือ เส้นทางเพื่อบอกตำแหน่งของไฟล์ หรือ directory

Root Directory คือ directory หลัก ที่รวม directory ทั้งหมด

Working Directory คือ directory ที่โปรแกรมเริ่มทำงาน

File System

Current Directory คือ directory ปัจจุบันที่อ้างอิงจากไฟล์ (.)

Parent Directory คือ directory ที่บรรจุ current directory (..)

File Separator คือ สัญลักษณ์ที่ใช้แบ่งชื่อไฟล์ หรือ ไดเรคทอรี ใน path

- Linux, Mac ใช้ / เช่น `/home/file.txt`
- Windows ใช้ \ เช่น `C:\Users\PC708`

File Name

- File Name คือ ชื่อไฟล์ รวมนามสกุลด้วย

  นามสกุลของไฟล์ จะอยู่หลัง . ทางขวาสุด

  โปรแกรมเมอร์มีสิทธิ์ตั้งนามสกุลของไฟล์ได้อิสระ

  แต่ควรใช้ตามมาตรฐาน

File v.s. Directory

- บางไฟล์ อาจไม่มีนามสกุล ทำให้ดูคล้าย directory
  (นิยมใช้ในตระกูล Linux)
- ถ้าเป็นชื่อ directory โดยทั่วไปนิยมใส่ Separator ต่อท้าย Path

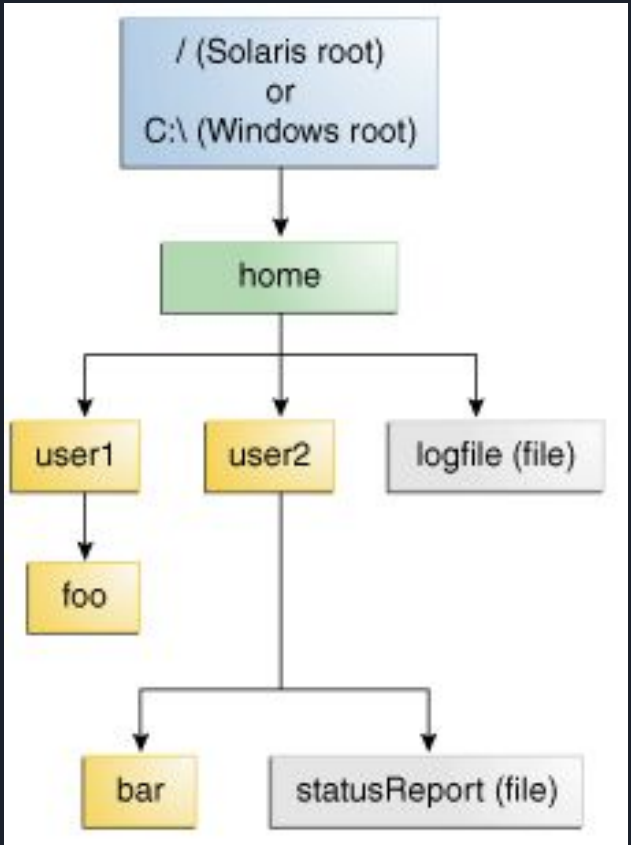
ถ้าเขียน Path `C:\Users\PC708`

จะเข้าใจได้ว่า PC708 เป็นไฟล์

ถ้าเขียน Path `C:\Users\PC708\`

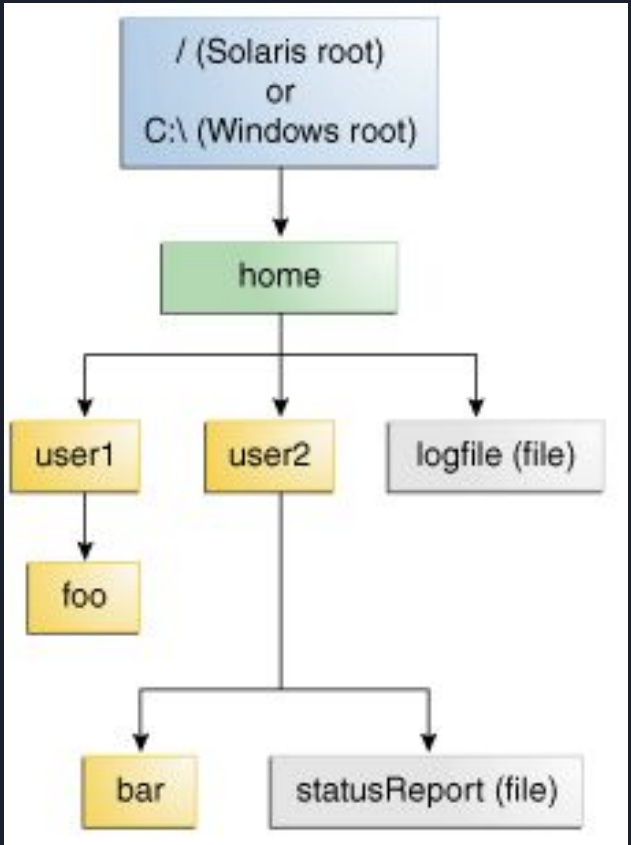
จะเข้าใจได้ว่า PC708 เป็น directory

Directory Tree




```
/
|
|--- home/
       |
       |--- user1/
       |      |
       |      |--- foo/
       |
       |--- user2/
       |      |
       |      |--- bar/
       |      |
       |      |--- statusReport
       |
       |--- logfile
```

Absolute Path



- Path ที่อ้างอิงจาก Root Directory
- Linux, Mac อ้างอิงจาก /
- Windows อ้างอิงจาก Drive Letter เช่น C:\

ถ้าต้องการอ้างอิงที่อยู่ไฟล์ statusReport

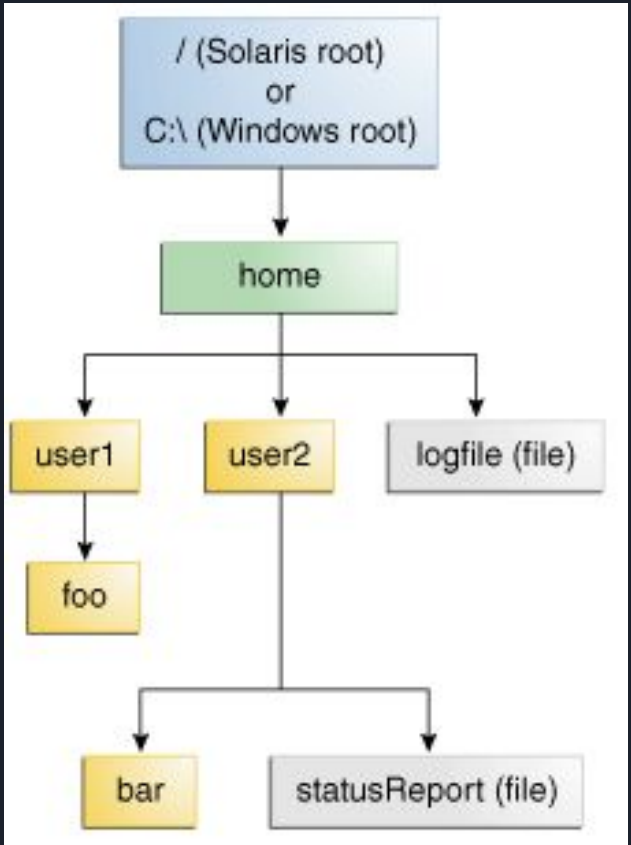
Linux, Mac:

```
/home/user2/statusReport
```

Windows:

```
C:\home\user2\statusReport
```

Relative Path



- Path ที่อ้างอิงจาก Current Directory หรือจาก Working Directory

ถ้าต้องการอ้างอิงที่อยู่ไฟล์ statusReport จาก directory foo/

Linux, Mac:

```
../../user2/statusReport
```

Windows

```
..\..\user2\statusReport
```

Absolute Path v.s. Relative Path

- การอ้างอิงแบบ Absolute Path เมื่อรันต่างเครื่องกัน

  จะต้องมี directory tree จาก root directory ที่เหมือนกันเท่านั้น จึงจะอ้างอิงได้

- การอ้างอิงแบบ Relative Path เมื่อรันต่างเครื่องกัน

  จะต้องมี directory tree เฉพาะส่วนที่อ้างอิงที่เหมือนกัน จึงจะอ้างอิงได้

- การอ้างอิงแบบ Relative Path อาจเกิดความสับสนระหว่างการอ้างอิงจาก current directory หรือ working directory

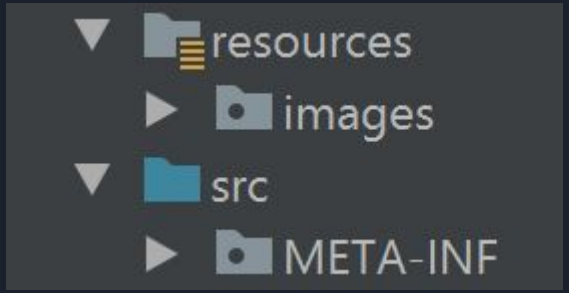
  ซึ่งบางสภาพแวดล้อมของการทำงาน current directory กับ working directory อาจไม่ใช่ตำแหน่งเดียวกัน

- แล้ว ควรเลือกใช้ Absolute Path หรือ Relative Path?

การจัดการไฟล์ภายในโปรเจค

https://dzone.com/articles/an-introduction-to-the-java-collections-framework

Resources Root

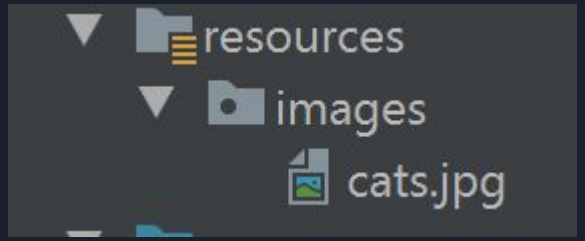


1. สร้าง directory ระดับเดียวกับ src

2. กำหนดให้ directory นั้นเป็น Resources Root
   โดยการคลิกขวาที่ directory นั้น แล้วเลือก
   Mark Directory as > Resources Root

การอ้างอิงผ่าน Resources Root

- ใช้เรียกทรัพยากรที่กำหนดไว้ในโปรเจคแล้ว เช่น รูปภาพ
- ไม่ต้องระบุชื่อ Resources Root



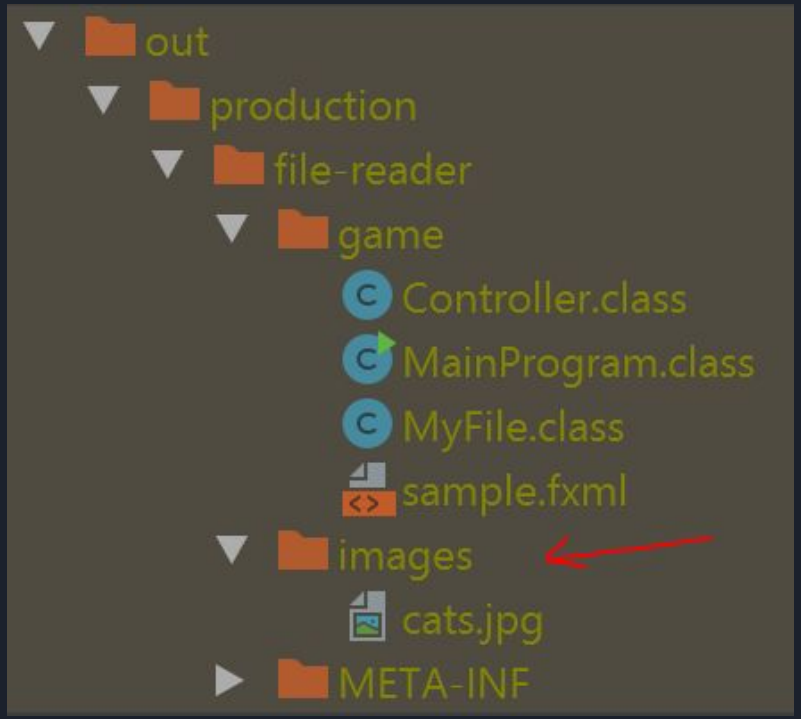

```
public class Controller {
    @FXML ImageView image;
    @FXML public void initialize() {
        image.setImage(new Image("images/cats.jpg"));
    }
}
```

Compile and Build

หลังการ compile

ไฟล์ที่อยู่ใน Resources Root จะไปอยู่ที่ Root ของ Project

โดยไม่มีชื่อของ directory ที่เป็น resources root



ไฟล์ภายในโปรเจค

อ่านได้

แต่เขียนไม่ได้

อยากเขียนได้ ต้องจัดการไฟล์แบบภายนอกโปรเจค

การจัดการไฟล์ภายนอกโปรเจค

https://dzone.com/articles/an-introduction-to-the-java-collections-framework

Temporary Directory

- ไฟล์ที่ถูกสร้างเพื่ออ่านหรือเขียน และเป็นไฟล์ภายนอกโปรเจค เมื่ออยู่ใน Intellij IDEA จะถูกสร้าง ณ ตำแหน่งเดียวกับ src/ (ไม่ใช่ใน src/)
- แต่ถ้ารันจาก executable jar file แล้ว ไฟล์ภายนอกโปรเจค จะถูกสร้าง ณ ตำแหน่งเดียวกับ jar file
- ควรสร้าง Temporary Directory เพื่อจัดการไฟล์เป็นสัดส่วน

Java File API

- java.io.File
- ใช้จัดการไฟล์ภายนอกโปรเจค
- จัดการทั้งไฟล์และ directory

IOException

```
java.io.IOException
    FileNotFoundException
    EOFException (End of File)
```

- เป็น Checked Exception
- ต้องจัดการด้วย try/catch เสมอ

การสร้างไฟล์

```
File file = new File("tmp/test.txt");

file.createNewFile(); // return boolean
```

* ถ้าไม่มี directory tmp จะเกิด Exception

* ถ้าไม่มีสิทธิ์ write ไฟล์ ใน directory จะสร้างไฟล์ไม่ได้

  (Linux และ Mac ต้องมีการจัดการ file permissions)

การสร้าง directory

```
File file = new File("tmp/sub-tmp");

file.mkdirs(); // return boolean
```

* tmp และ sub-tmp ถูกมองเป็น directory

* ชื่อ directory มี . ได้

File System Properties (user directory)

- แก้ปัญหา current directory v.s. working directory
- โดยการกำหนด Current Path จาก root directory

```
String cwd =
        System.getProperty("user.dir");
```

File System Properties (file separator)

```
String FS =
    System.getProperty("file.separator");

FS = File.separator;
```

File Status

```
file.exists()

file.isFile()

file.isDirectory()

file.isHidden()
```

return boolean

Read File (BufferedReader)

```
File file = new File("test.txt");

FileReader fileReader = new FileReader(file);

BufferedReader reader = new BufferedReader(fileReader);

String line = "";

while ((line = reader.readLine()) != null) {

    // ดำเนินการทีละบรรทัดกับตัวแปร line

}

reader.close();
```

Write File (BufferedWriter)

```
File file = new File("test.txt");

FileWriter fileWriter = new FileWriter(file);

BufferedWriter writer = new BufferedWriter(fileWriter);

writer.write("data");

writer.newLine();

writer.append("text");

writer.close();
```

Write File Append Parameter

```
FileWriter fileWriter =
     new FileWriter(file[, append]);
```

- ถ้ากำหนดเป็น true เมื่อเรียก method write()

  จะเป็นการเขียนต่อจากเนื้อหาในไฟล์เดิม

ตัวอย่าง

https://github.com/saacsos/javafx-file-reader-writer