Tiếng nói của chuyên gia về Công nghệ Java

Bí mật Java 7

Tiết lộ những chức năng mới trong Java 7

Phan Thành Nhân

Nội dung

CHAPTER 0: GIỚI THIỆU 3
CHAPTER 1: JAVA LANGUAGE IMPROVEMENTS 4
I. Sử dụng String trong switch/case 4
II. Sử dụng gạch dưới trong literal 5
III. Sử dụng try-with-resources 5
IV. Tạo resource để sử dụng với try-with-resources 6
V. Catch nhiều exception 7
VI. Re-throw exception 7
VII. Diamond operator 7
VIII. Sử dụng annotation @SafeVarargs 8
CHAPTER 2: NEW FILESYSTEM API NIO.2 9
I. Path 9
1. Tạo Path object 9
2. Tương tác giữa java.io.File and java.nio.fle.Files 11
3. Converting path tương đối sang path tuyệt đối 12
4. Xóa bỏ dư thừa bằng cách normalize path 13
5. Nối 2 paths 13
II. Một số tác vụ khác với NIO.2 14
CHAPTER 3: THEO DÕI DIRECTORY 15

CHAPTER 0: GIỚI THIỆU

Ebook này tôi sẽ trình bày sơ qua một số điểm mới nổi bật của Java 7, không như Ebook về Java 6, tôi sẽ không đi sâu vào phân tích sự thay đổi từng package mà sẽ chỉ giới thiệu những đổi mới về mặt ngôn ngữ, một ít API mới và demo cách dùng mà thôi. Nếu muốn biết nhiều và sâu hơn thì bạn hãy tìm đồ chơi mới trên Google nhé =))

CHAPTER 1: JAVA LANGUAGE IMPROVEMENTS

Java 7 được release vào 7/2011 và ra mắt khá nhiều feature mới. Chapter này sẽ tập trung vào những điểm mới được giới thiệu trong Project Coin. Project Coin bao gồm những thay đổi nhỏ trong Java 7 để hướng tới việc giảm thiểu những đoạn code dư thừa, làm cho chương trình dễ đọc hơn

Trong phần này, chúng ta sẽ tìm hiểu những điểm sau:

- Sử dụng string trong switch/case
- Sử dụng dấu gạch dưới cho literal để tăng readability
- Sử dụng try-with-resources để improve exception handling
- Tạo resource để sử dụng với try-with-resources
- Catch nhiều exception
- Re-throw exception
- Sử dụng diamond operator
- Sử dụng annotation @SafeVarargs

I. Sử dụng String trong switch/case

Trước đây, switch/case chỉ dùng được với giá trị integer. Nhưng việc switch giữa những giá trị chuỗi là khá phổ biến, nó làm cho code dễ đọc hơn rất nhiều.

Java 7 đã hỗ trợ switch trên string, nhưng có một điều thú vị rằng Java Virtual Machine (JVM) vẫn chưa hỗ trợ switch với string, Java compiler chịu trách nhiệm convert string trong switch sang byte code một cách thích hợp để chương trình chạy được.

Có 2 điều cần lưu ý:

- Việc switch trên biến null sẽ quăng NullPointerException
- Các case của switch là case sensitive

II. Sử dụng gạch dưới trong literal

Numberical literal trong Java 7 có thể chứa các ký tự gạch dưới (_), điều này làm cho code dễ đọc hơn vì lúc này một chuỗi dài các số có thể được gom lại thành các nhóm. Hãy xem ví dụ như sau

int x = 000000000000;

int x = 000_000_000_000;

Ở ví dụ dưới, người đọc code sẽ thấy rõ ràng 12 số 0, trường hợp ở trên phải căng mắt ra đếm thì hơi fail =)) Giá trị của 2 trường hợp đều như nhau.

Cách này có thể áp dụng cho các primitive type như binary, octal, hexadecimal, hay decimal, integer hoặc floating-point literal.

Những trường hợp KHÔNG thể dùng

- Sử dụng underscore ở đầu hoặc cuối số: _12345_67890_09876_54321L;
- Sử dụng underscore gần với dấu chấm khi chơi với float, double: 3._14_15F;
- Trước những hậu tố D, F, L hoặc những hậu tố string: 123_456_789_L;

III. Sử dụng try-with-resources

Trước Java 7, những đoạn code sử dụng resource yêu cầu thao tác mở và đóng, ví dụ như java.io.InputStream, java.nio.Channel, việc đóng resource rất dễ bị quên dẫn đến lãng phí tài nguyên. Try-with-resources bock được thêm vào Java 7 như là cách để đơn giản hóa xử lý error và làm ngắn gọn code.

```
Try (Resource1 res1 = new Resource1(); Resource2 res2 = new
Resource2()) {

}
```

Sau khi try kết thúc, res1 và res2 sẽ tự động đóng. Để làm được điều này thì class Resource1 và Resource2 phải implements java.lang.AutoCloseable.

IV. Tạo resource để sử dụng với try-with-resources

Java có khá nhiều resource để chơi với try-with-resources, nhưng bạn vẫn có thể tạo resource của riêng bạn để sử dụng với cái này bằng cách implement java.lang.AutoCloseable, interface này chỉ gồm 1 method là close().

```
import java.lang.AutoCloseable;
public MyResource implements AutoCloseable {
      @Override
      public void close() {
            System.out.println(“Resource closed!”);
      }
}

public class Main {
      public static void main(String[] args) {
            try (MyResource resource = new MyResource()) {
                  System.out.println(“Resource opened!”);
            } catch …
      }
}
```

V. Catch nhiều exception

Java 7 đã có thể handle nhiều exception trong 1 catch, mục tiêu của feature này là reduce the duplication of code.

```
catch (InputMismatchException | InvalidParameter e) {
      ...
}
```

Catch nhiều exception trong 1 catch block giúp giảm trùng lặp code, nhưng nếu những exception đó có cùng 1 base class thì nên catch base class thay vì nhiều cái. Ví dụ, catch java.io.IOException thay vì java.nio.file.NoSuchFileException và java.nio.file.DirectoryNotEmptyException

Java 7 có thêm 1 exception ReflectiveOperationException là base class của ClassNotFoundException, IllegalAccessException, InstantiationException, InvocationTargetException, NoSuchFieldException, NoSuchMethodException

Chức năng mới này hướng tới việc handle những exception không có cùng base class.

VI. Re-throw exception

VII. Diamond operator

Sử dụng <>, không cần phải chỉ rõ lại type nữa

List<String> list = new ArrayList<>();

VIII. Sử dụng annotation @SafeVarargs

```
public <T> void foo(T… params) {
      for (T t : params) {
            System.out.println(t.toString());
      }
}
```

Ví dụ trên sử dụng varargs với generic type, Java 7 sẽ warning "Type safety: Potential heap pollution via varargs parameter params". Sử dụng @SafeVarargs để xác định rằng việc sử dụng ở đây là an toàn.

Để sử dụng @SafeVarargs thì method foo phải chuyển sang final hoặc static.

```
@SafeVarargs
public final <T> void foo(T… params) {
      for (T t : params) {
            System.out.println(t.toString());
      }
}
```

@SafeVarargs sử dụng để thông báo compiler rằng method này sẽ không dẫn đến heap pollution, do đó phải chắc rằng method đó không dẫn đến heap pollution mới dùng @SafeVarargs, nếu không thì lúc chạy sẽ gặp error.

CHAPTER 2: NEW FILESYSTEM API NIO.2

Trước đây, java.io là công cụ duy nhất để làm việc với file, api này tồn tại một số hạn chế như thiếu method copy file, nhiều method return boolean mà không quăng exception nên việc debug khó khăn, … Java.nio ra đời ở bản java 1.4 khắc phục những hạn chế này, nhưng chủ yếu hỗ trợ các tác vụ low level. Do đó, Java 7 giới thiệu NIO.2 bổ sung cho nio với các chức năng quản lý hệ thống file. Sau đây sẽ là giới thiệu sơ qua về Path mới và một số thao tác với NIO.2

I. Path

1. Tạo Path object

Path dùng để xác định resource, một số cách để tạo path như sau:

- FileSystem. getPath()
- Paths.get()

```
Path path = FileSystems.getDefault().getPath("/home/docs/
status.txt");
System.out.println();
System.out.printf("toString: %s\n", path.toString());
System.out.printf("getFileName: %s\n", path.getFileName());
System.out.printf("getRoot: %s\n", path.getRoot());
System.out.printf("getNameCount: %d\n", path.getNameCount());
for(int index=0; index<path.getNameCount(); index++) {
System.out.printf("getName(%d): %s\n", index, path.
   getName(index));
   }
System.out.printf("subpath(0,2): %s\n", path.subpath(0, 2));
System.out.printf("getParent: %s\n", path.getParent());
System.out.println(path.isAbsolute());
```

Output:

```
toString: \home\docs\status.txt
getFileName: status.txt
getRoot: \
getNameCount: 3
getName(0): home
getName(1): docs
getName(2): status.txt
subpath(0,2): home\docs
getParent: \home\docs
false
```

Paths.get() cũng có thể dùng để tạo Path object

```
try {
      path = Paths.get("/home", "docs", "users.txt");
      System.out.printf("Absolute path: %s", path.
      toAbsolutePath());
}
catch (InvalidPathException ex) {
      System.out.printf("Bad path: [%s] at position %s",
      ex.getInput(), ex.getIndex());
}
```

Ví dụ trên sử dụng "/home", Path được tạo sẽ bắt đầu từ ổ đĩa đang làm việc, ví dụ C:/home/docs/users.txt. Trường hợp sử dụng "home" thì sẽ bắt đầu từ thư mục đang làm việc, ví dụ C:/WorkSpace/Java7/home/docs/users.txt

2. Tương tác giữa java.io.File and java.nio.fle.Files

Trước khi java.nio được giới thiệu, java.io là cách duy nhất để làm việc với file và thư mục. Cho dù nio ra đời hỗ trợ hầu như tất cả những gì io có thể làm được, nhưng việc sử dụng class cũ của io vẫn rất ok.

Bạn có thể tạo Path object từ java.io.File object sử dụng File.toPath()

```
    public static void main(String[] args) {
          try {
                Path path =
                Paths.get(new
                URI("file:///C:/home/docs/users.txt"));
                File file = new File("C:\\home\\docs\\users.txt");
                Path toPath = file.toPath();
                System.out.println(toPath.equals(path)); // true
          } catch (URISyntaxException e) {
                System.out.println("Bad URI");
          }
    }
```

3. Converting path tương đối sang path tuyệt đối

Path có path tuyệt đối và path tương đối, cả 2 được sử dụng trong những trường hợp nhất định. Path tương đối sử dụng để xác định location của file hoặc thư mục trong mối quan hệ với thư mục hiện hành. Path tuyệt đối là path đi từ root level, mỗi thư mục cách nhau bằng dấu slash

```
String separator = FileSystems.getDefault(). getSeparator();
System.out.println("The separator is " + separator);
try {
      Path path = Paths.get(new URI("file:///C:/home/docs/
      users.txt"));
      System.out.println("subpath: " + path.subpath(0, 3));
      path = Paths.get("/home", "docs", "users.txt");
      System.out.println("Absolute path: " +
      path.toAbsolutePath());
      System.out.println("URI: " + path.toUri());
}
catch (URISyntaxException ex) {
      System.out.println("Bad URI");
}
catch (InvalidPathException ex) {
      System.out.println("Bad path: [" + ex.getInput() + "] at
      position
      " + ex.getIndex());
}
```

Output:

The separator is \

subpath: home\docs\users.txt

Absolute path: E:\home\docs\users.txt

URI: fle:///E:/home/docs/users.txt

4. Xóa bỏ dư thừa bằng cách normalize path

Xét thư mục home có 2 thư mục con là docs và music, docs có 1 file là docs.txt, music có 1 file là music.txt

Có 2 đường dẫn như sau:

/home/docs/../music/music.txt

/home/docs/./docs.txt

Path thứ 1 dư /docs/..

Path thứ 2 dư /.

Để xóa bỏ sự dư thừa này, gọi path.normalize()

5. Nối 2 paths

Path rootPath = Paths.get("/home/docs");

Path partialPath = Paths.get("users.txt");

Path resolvedPath = rootPath.resolve(partialPath); // \home\docs\users.txt

II. Một số tác vụ khác với NIO.2

```
// Copy file
try {
	Path source = Paths.get("/Users/admin/Desktop/Java/in.txt");
	Path dest = Paths.get("/Users/admin/Desktop/Java/out.txt");
	Files.copy(source, dest);
} catch (IOException e) {
	e.printStackTrace();
}
// Check if file exist
Path p = Paths.get(HOME);
Files.exists(p);
Files.notExists(p);

// Check file hay directory
Files.isRegularFile(p);

// Check file permissions
Files.isReadable(p);
Files.isWritable(p);
Files.isExecutable(p);

// Check same files
Files.isSameFile(p1, p2);

// Create file, delete file, move file ...
```

CHAPTER 3: THEO DÕI DIRECTORY

Java 7 cung cấp công cụ để theo dõi các event của directory, bao gồm các event THÊM MỚI, MODIFY, XÓA

Ví dụ sau demo việc theo dõi các event của thư mục Java ngoài Desktop

```
try {
      WatchService watchService =
      FileSystems.getDefault().newWatchService();
      Path path = Paths.get("/Users/admin/Desktop/Java/");
      path.register(watchService,
          StandardWatchEventKinds.ENTRY_CREATE,
          StandardWatchEventKinds.ENTRY_MODIFY,
          StandardWatchEventKinds.ENTRY_DELETE);
      WatchKey key = null;
      while (true) {
            key = watchService.take();
            for (WatchEvent<?> event : key.pollEvents()) {
                  WatchEvent.Kind<?> kind = event.kind();
                  System.out.println("Event on "+
                  event.context().toString() + " is " + kind);
            }
            boolean reset = key.reset();
            if (!reset) {
                  break;
            }
      }
} catch (IOException e) {
      e.printStackTrace();
} catch (InterruptedException e) {
      e.printStackTrace();
}
```