Câu hỏi JFrame.dispose () vs System.exit ()

Sự khác biệt giữa hai phương pháp này là gì - `System.exit()` và `JFrame.dispose()` ?

Nếu chúng ta muốn đóng một ứng dụng Java Swing khi một nút được bấm, phương pháp nào tôi nên sử dụng?

30

2017-11-13 12:03

gốc

> trên nút bấm. là về JButton độc lập với System.exit (0) hoặc từ nút trong JFrames ToolBar - mKorbel
>
> @mKorbel: tùy chọn đầu tiên - Adil

Các câu trả lời:

`System.exit();` làm cho máy ảo Java kết thúc hoàn toàn.

`JFrame.dispose();` gây ra `JFrame` cửa sổ bị hủy và dọn dẹp bởi hệ điều hành. Theo tài liệu, điều này có thể khiến máy ảo Java chấm dứt nếu không có Windows nào khác, nhưng điều này thực sự chỉ được xem như là một hiệu ứng phụ chứ không phải là chuẩn.

Cái bạn chọn thực sự phụ thuộc vào tình huống của bạn. Nếu bạn muốn chấm dứt mọi thứ trong máy ảo Java hiện tại, bạn nên sử dụng `System.exit()` và mọi thứ sẽ được dọn sạch. Nếu bạn chỉ muốn phá hủy cửa sổ hiện tại, với hiệu ứng phụ, nó sẽ đóng máy ảo Java nếu đây là cửa sổ duy nhất, sau đó sử dụng `JFrame.dispose()` .

49

2017-11-13 12:08

> Vấn đề tôi thấy với System.exit là nó thực sự giết chết ứng dụng, ngay cả khi một sợi là, ví dụ, viết một tập tin. Dường như với tôi rằng hầu hết các ứng dụng đóng cửa sổ, hơn là đợi phần còn lại của các chủ đề hoàn thành. Nhưng khi các chủ đề AWT có thể không hoàn thành, chúng tôi đã có một vấn đề làm thế nào để thực hiện điều này trong Java đúng cách. - Tomáš Zato

JFrame.dispose ()

```
public void dispose()
```

Câu hỏi phổ biến

Danh sách hình ảnh "chế độ"

Làm thế nào để xoay văn bản trong WPF bằng cách giữ chức năng Auto-Sizing

Nên package-lock.json cũng được công bố?

Gọi-Expression với exe trong Program Files

Tại sao rõ ràng std :: di chuyển là cần thiết khi trở về loại tương thích?

Kích thước phông chữ giảm khi xuất Crystal Report sang PDF

Thiết kế: Khi đường giữa các đối tượng miền và đối tượng dịch vụ không rõ ràng

Phát hành tất cả các tài nguyên màn hình nguyên bản được sử dụng bởi Cửa sổ này, các thành phần con của nó và tất cả các con của nó. Nghĩa là, tài nguyên cho các thành phần này sẽ bị hủy, mọi bộ nhớ mà chúng tiêu thụ sẽ được trả lại cho hệ điều hành và chúng sẽ được đánh dấu là không thể phát được. Cửa sổ và các thành phần con của nó có thể được hiển thị lại bằng cách xây dựng lại các tài nguyên gốc với một cuộc gọi tiếp theo để đóng gói hoặc hiển thị. Các trạng thái của cửa sổ được tái tạo và các thành phần con của nó sẽ giống hệt với trạng thái của các đối tượng này tại điểm mà Cửa sổ đã được xử lý (không tính đến các sửa đổi bổ sung giữa các hành động đó).

Chú thích: Khi cửa sổ hiển thị cuối cùng trong máy ảo Java (VM) được xử lý, máy ảo có thể chấm dứt. Xem Các vấn đề về phân luồng AWT để biết thêm thông tin.

System.exit ()

```
public static void exit(int status)
```

Chấm dứt Java Virtual Machine hiện đang chạy. Đối số đóng vai trò như một mã trạng thái; theo quy ước, mã trạng thái không đồng bộ cho biết chấm dứt bất thường. Phương thức này gọi phương thức exit trong lớp Runtime. Phương pháp này không bao giờ trở lại bình thường.

Cuộc gọi `System.exit(n)` có hiệu quả tương đương với cuộc gọi:

```
Runtime.getRuntime().exit(n)
```

11

2017-11-13 12:09

Ngoài những điều trên, bạn có thể sử dụng `System.exit()` để trả lại mã thoát có thể rất đặc biệt nếu bạn gọi quá trình tự động bằng cách sử dụng `System.exit(code);` điều này có thể giúp bạn xác định ví dụ nếu một lỗi đã xảy ra trong khi chạy.

5

2017-11-13 12:12

- Nếu bạn có nhiều cửa sổ đang mở và chỉ muốn đóng cửa sổ đã bị đóng `JFrame.dispose()`.
- Nếu bạn muốn đóng tất cả các cửa sổ và chấm dứt việc sử dụng ứng dụng `System.exit()`

3

2017-08-12 23:17