



Love beautiful code? We do too.

Home / Tutorial / Học Java / Sử dụng Iterator trong Java

- Tất cả
- My Feed
- Đã giải quyết
- Danh mục

Danh mục của bạn

- Html
- CSS
- Javascript
- PHP
- Jquery
- Laravel
- AngularJS
- NodeJS
- Java
- Python
- VueJS
- MySQL
- C
- Swift
- React Native
- MongoDB
- Docker

« Previous Lesson

Next Lesson »

Sử dụng Iterator trong Java

Sử dụng Iterator trong Java, Thường thì, bạn sẽ muốn tuần hoàn qua các phần tử trong một tập hợp. Ví dụ, bạn có thể muốn hiển thị mỗi phần tử.

Cách đơn giản nhất để thực hiện điều này là thuê một Iterator, là một đối tượng mà triển khai hoặc Iterator hoặc ListIterator interface.

Iterator trong Java cho bạn khả năng để tuần hoàn qua một tập hợp, kiếm được và gỡ bỏ các phần tử. ListIterator kế thừa Iterator để cho phép "vọc" song hướng một danh sách và sửa đổi các phần tử.

Trước khi bạn có thể truy cập một Collection thông qua một Iterator, bạn phải có được nó. Mỗi lớp Collection cung cấp một phương thức **iterator()** mà trả về một iterator tới phần bắt đầu của Collection. Bởi sử dụng đối tượng Iterator, bạn có thể truy cập mỗi phần tử trong Collection, từng phần tử một tại một thời điểm.

Để hiểu sâu hơn các khái niệm được trình bày trong chương này, mời bạn tham khảo loạt bài:

Nói chung, để sử dụng một iterator để tuần hoàn qua các nội dung của một Collection, bạn theo các bước sau:

- Đặt được một iterator tới phần đầu của Collection bằng cách gọi phương thức `iterator()` của Collection trong Java.
- Thiết lập một vòng lặp mà tạo triệu hồi tới `hasNext()`. Vòng lặp này lặp đi lặp lại tới khi `hasNext()` trả về true.
- Trong vòng lặp, thu được mỗi phần tử bởi triệu hồi phương thức `next()`.

Với các Collection mà triển khai List, bạn cũng có thể thu được một iterator bởi triệu hồi `ListIterator`.

Phương thức được khai báo bởi Iterator trong Java

STT	Phương thức và Miêu tả
1	boolean hasNext() Trả về true nếu có nhiều phần tử. Nếu không là false
2	Object next() Trả về phần tử kế tiếp. Ném <code>NoSuchElementException</code> nếu không có một phần tử kế tiếp
3	void remove()

Danh mục bài học

JAVA CƠ BẢN

- Mở đầu
- Java là gì?
- Lịch sử Java
- Tổng quan về Java
- Cài đặt môi trường Java
- Cách thiết lập PATH trong Java
- Chương trình Java đầu tiên Hello World
- Phân tích nội tại chương trình Hello World trong Java
- Cú pháp Java cơ bản
- Giới thiệu JDK, JRE và JVM trong Java
- Các kiểu biến trong Java
- Kiểu dữ liệu trong Java
- Toán tử trong Java
- Vòng lặp trong Java
- Lệnh IF/ELSE, Lệnh SWITCH/CASE trong Java
- Number trong Java
- Character trong Java

KHÁI NIỆM HƯỚNG ĐỐI TƯỢNG

- Khái niệm hướng đối tượng (OOP) trong Java
- Đối tượng và lớp (class) trong Java
- Phương thức trong Java
- Nạp chồng phương thức trong Java

Gỡ bỏ phần tử hiện tại. Ném `IllegalStateException` nếu cố gắng gọi phương thức `remove()` mà không được đặt trước một triệu hồi tới `next()`

Phương thức được khai báo bởi `ListIterator` trong Java

STT	Phương thức và Miêu tả
1	<code>void add(Object obj)</code> Chèn <code>obj</code> vào trong <code>List</code> ở trước phần tử mà sẽ được trả về bởi lần triệu hồi tiếp theo tới <code>next()</code>
2	<code>boolean hasNext()</code> Trả về <code>true</code> nếu có một phần tử kế tiếp. Nếu không là <code>false</code>
3	<code>boolean hasPrevious()</code> Trả về <code>true</code> nếu có một phần tử ở trước. Nếu không là <code>false</code>
4	<code>Object next()</code> Trả về phần tử kế tiếp. Ném <code>NoSuchElementException</code> nếu không có phần tử đó
5	<code>int nextIndex()</code> Trả về chỉ mục của phần tử kế tiếp. Nếu không có phần tử này, trả về kích cỡ của <code>list</code>
6	<code>Object previous()</code> Trả về phần tử trước. Ném <code>NoSuchElementException</code> nếu không có phần tử đó
7	<code>int previousIndex()</code> Trả về chỉ mục của phần tử ở trước. Nếu không có phần tử này, trả về -1
8	<code>void remove()</code> Gỡ bỏ phần tử hiện tại từ <code>list</code> . Ném <code>IllegalStateException</code> nếu <code>remove()</code> được triệu hồi trước khi <code>next()</code> hoặc <code>previous()</code> được gọi
9	<code>void set(Object obj)</code> Gán <code>obj</code> tới phần tử hiện tại. Đây là phần tử cuối cùng được trả về bởi một triệu hồi tới hoặc <code>next()</code> hoặc <code>previous()</code>

Ví dụ

Sau đây là ví dụ minh họa cả `Iterator` và `ListIterator`. Nó sử dụng một đối tượng `ArrayList`, nhưng các qui tắc chung áp dụng tới bất kỳ kiểu `Collection` nào.

Tất nhiên, `ListIterator` chỉ có sẵn cho các `Collection` mà triển khai `List` Interface trong Java:

```
import java.util.*;

public class IteratorDemo {

    public static void main(String args[]) {
        // Tao mot array list
        ArrayList al = new ArrayList();
        // them cac phan tu toi array list
```

• Tập chung phương thức trong Java

• Constructor trong Java

• Từ khóa `static` trong Java

• Từ khóa `this` trong Java

• Tính kế thừa trong Java - Từ khóa `extends` và `implements` trong Java

• Quan hệ HAS-A trong Java

• Ghi đè phương thức trong Java

• Kiểu trả về `covariant` trong Java

• Từ khóa `super` trong Java

• Từ khóa `final` trong Java

• Đa hình trong Java

• Gắn kết tĩnh và Gắn kết động (`Dynamic Binding`) trong Java

• Toán tử `instanceof` trong Java

• Tính trừu tượng trong Java

• Lớp trừu tượng - `Abstract Class` trong Java

• Interface trong Java

• Phân biệt lớp `abstract` và `Interface` trong Java

• Package trong Java

• Các kiểu `Modifier` trong Java

• Non Access `Modifier` trong Java

• Access `Modifier` trong Java

• Tính bao đóng trong Java

• Lớp `Object` trong Java

• Nhân bản đối tượng trong Java

• Mảng (`Array`) trong Java

• Lớp `Wrapper` trong Java

• Gọi bởi giá trị trong Java

• Từ khóa `strictfp` trong Java

• Date và Time trong Java

```
al.add("C");
al.add("A");
al.add("E");
al.add("B");
al.add("D");
al.add("F");

// Su dung iterator de hien thi noi dung cua array list
System.out.print("Noi dung ban dau cua ArrayList la: ");
Iterator itr = al.iterator();
while(itr.hasNext()) {
    Object element = itr.next();
    System.out.print(element + " ");
}
System.out.println();

// Sua doi cac doi tuong sau khi da duoc lap qua
ListIterator litr = al.listIterator();
while(litr.hasNext()) {
    Object element = litr.next();
    litr.set(element + "+");
}
System.out.print("Noi dung sau khi sua doi cua ArrayList la: ");
itr = al.iterator();
while(itr.hasNext()) {
    Object element = itr.next();
    System.out.print(element + " ");
}
System.out.println();

// Bay gio, hien thi list theo chieu nguoc lai
System.out.print("Hien thi list theo chieu nguoc lai: ");
while(litr.hasPrevious()) {
    Object element = litr.previous();
    System.out.print(element + " ");
}
System.out.println();
}
```

Nó sẽ cho kết quả sau:

```
Noi dung ban dau cua ArrayList la: C A E B D F
Noi dung sau khi sua doi cua ArrayList la: C+ A+ E+ B+ D+ F+
Hien thi list theo chieu nguoc lai: F+ D+ B+ E+ A+ C+
```

↑ 0

💬 0

🔍 📌 🔄

« Previous Lesson

Next Lesson »

🗨️ Viêt câu trả lời

👁️ Preview

Drop Images

B I H S | </> “ ” ☰ ☷ 🔗 🖼️ 📄 👁️ 🗑️ ❌ ?

Nội dung câu hỏi.. [Markdown Supported]

⊕ Regular Expression trong Java
FILE VÀ I/O TRONG JAVA
⊕ File và I/O trong Java
⊕ ByteArrayInputStream trong Java
⊕ DataInputStream trong Java
⊕ ByteArrayOutputStream
⊕ DataOutputStream trong Java
⊕ Lớp File trong Java
⊕ Lớp FileReader trong Java
⊕ Lớp FileWriter trong Java
STRING TRONG JAVA
⊕ String trong Java
⊕ Immutable String trong Java
⊕ So sánh chuỗi trong Java
⊕ Nối chuỗi trong Java
⊕ Chuỗi con trong Java
⊕ Phương thức của lớp String trong Java
⊕ Lớp StringBuffer trong Java
⊕ StringBuilder trong Java
⊕ So sánh lớp String và StringBuffer trong Java
⊕ So sánh lớp StringBuffer và StringBuilder trong Java
⊕ Tạo lớp Immutable trong Java
⊕ String toString() trong Java
⊕ Lớp StringTokenizer trong Java
XỬ LÝ NGOẠI LỆ (EXCEPTION HANDLING)
⊕ Exception trong Java
⊕ Khối try-catch trong Java
⊕ Khối finally trong Java

lines: 1 words: 0 0.0 2 Ke



✚ Từ khóa throw trong Java

✚ ExceptionHandling và Ghi đè phương thức trong Java

✚ Custom Exception trong Java

CẤU TRÚC DỮ LIỆU TRONG JAVA

✚ Cấu trúc dữ liệu trong Java

✚ Enum trong Java

✚ Lớp BitSet trong Java

✚ Lớp Vector trong Java

✚ Lớp Stack trong Java

✚ Lớp Dictionary trong Java

✚ Lớp Hashtable trong Java

✚ Lớp Properties trong Java

COLLECTION TRONG JAVA

✚ Collection trong Java

✚ Collection Interface trong Java

✚ List Interface trong Java

✚ Set Interface trong Java

✚ SortedSet Interface trong Java

✚ Map Interface trong Java

✚ Map.Entry Interface trong Java

✚ SortedMap Interface trong Java

✚ Lớp LinkedList trong Java

✚ Lớp ArrayList trong Java

✚ Lớp HashSet trong Java

✚ Lớp LinkedHashSet trong Java

✚ Lớp TreeSet trong Java

✚ Lớp HashMap trong Java

✚ Lớp TreeMap trong Java

✚ Thuật toán Collection trong Java

✚ Sử dụng Iterator trong Java

➤ Sử dụng Comparator trong Java

JAVA NÂNG CAO

➤ Generic trong Java

➤ Serialization trong Java

➤ Lập trình mạng Socket - Lập trình mạng Socket trong Java

➤ Gửi Email trong Java

➤ Thread trong Java

➤ Cơ bản về Applet trong Java

➤ Tạo Javadoc

CHI TIẾT VỀ VÒNG LẶP TRONG JAVA

➤ Vòng lặp while trong Java

➤ Vòng lặp for trong Java

➤ Vòng lặp do...while trong Java

➤ Vòng lặp foreach trong Java

➤ Lệnh break trong Java

➤ Lệnh continue trong Java

ĐIỀU KHIỂN LƯỚING TRONG JAVA

➤ Lệnh if trong Java

➤ Lệnh if...else trong Java

➤ Lồng lệnh if trong Java

➤ Lệnh switch trong Java

TÀI LIỆU THAM KHẢO JAVA

➤ Tài liệu tham khảo về Java

Hoclaptrinh.vn © 2017

From Coder With ❤