HowToDoInJava

Java 8 forEach()

Last Updated: August 22, 2020 By: Lokesh Gupta Java 8 Java 8, Java Loops

The **Java forEach()** method is a utility function to iterate over a collection such as (list, set or map) and stream. It is used to perform a given action on each the element of the collection.

The `forEach()` method has been added in following places:

- **`Iterable` interface** – This makes `Iterable.forEach()` method available to all collection classes except `Map`
- **`Map`** interface – This makes `forEach()` operation available to all map classes.
- **`Stream`** interface – This makes `forEach()` and `forEachOrdered()` operations available to all types of stream.

1. Iterable forEach()

1.1. forEach() Method

The given code snippet shows the default implementation of `forEach()` method in Iterable interface.

Internally it uses the enhanced for-loop. So using the new for-loop will give the same effect and performance as `forEach()` method.

```
Iterable.java
```

```
default void forEach(Consumer<? super T> action)
{
    Objects.requireNonNull(action);
    for (T t : this) {
        action.accept(t);
    }
}
```

The `forEach()` method performs the given `action` for each element of the `Iterable` until all elements have been processed or the `action` throws an exception.

Example 1: Java program to iterate over a List using forEach()

```
Using forEach() method

List<String> names = Arrays.asList("Alex", "Brian", "Charles");

names.forEach(System.out::println);
```

Program Output:

```
Alex
Brian
Charles
```

1.2. Creating consumer action

In above example, the `action` represents an operation that accepts a single input argument and returns no result. It is an instance of `Consumer` interface.

By creating the consumer action like this, we can specify *multiple statements* to be executed in a syntax similar to a method.

```
Creating consumer action

List<String> names = Arrays.asList("Alex", "Brian", "Charles");

Consumer<String> makeUpperCase = new Consumer<String>()
{
    @Override
    public void accept(String t)
```

```
    {
      System.out.println(t.toUpperCase());
    }
};

names.forEach(makeUpperCase);
```

Program Output:

```
ALEX
BRIAN
CHARLES
```

2. Map forEach()

2.1. forEach() Method

This method performs the given BiConsumer action for each `Entry` in this `Map` until all entries have been processed or the action throws an exception.

```
Map.java

default void forEach(BiConsumer<? super K, ? super V> action) {
    Objects.requireNonNull(action);
    for (Map.Entry<K, V> entry : entrySet()) {
        K k;
        V v;
        try {
            k = entry.getKey();
            v = entry.getValue();
        } catch(IllegalStateException ise) {
            // this usually means the entry is no longer in the map.
            throw new ConcurrentModificationException(ise);
        }
        action.accept(k, v);
    }
}
```

Example 2: Java program to iterate over a Map using forEach()

```
Using Map.forEach() method
```

```
Map<String, String> map = new HashMap<String, String>();

map.put("A", "Alex");
map.put("B", "Brian");
map.put("C", "Charles");

map.forEach((k, v) ->
  System.out.println("Key = " + k + ", Value = " + v));
```

Program Output:

```
Key = A, Value = Alex
Key = B, Value = Brian
Key = C, Value = Charles
```

We can also create a *custom BiConsumer action* which will take key-value pairs from `Map` and process each entry one at a time.

```
Create custom BiConsumer

BiConsumer<String, Integer> action = (a, b) ->
{
  //Process the entry here as per business
    System.out.println("Key is : " + a);
    System.out.println("Value is : " + b);
};

Map<String, Integer> map = new HashMap<>();

map.put("A", 1);
map.put("B", 2);
map.put("C", 3);

map.forEach(action);
```

Program output.

```
Console

Key is : A
Value is : 1

Key is : B
```

```
Value is : 2

Key is : C
Value is : 3
```

3. Stream forEach() and forEachOrdered()

In `Stream`, `forEach()` and `forEachOrdered()` are terminal operations.

Similar to `Iterable`, stream `forEach()` method performs an action for each element of the stream.

For *sequential streams*, the order of elements (during iteration) is same as the order in the stream source, so the output would be same whether we use `forEach()` or `forEachOrdered()`.

while using *parallel streams*, use `forEachOrdered()` if order of the elements matter during the iteration. `forEach()` method does not gaurantee the element ordering to provide the advantages of parallelism.

Example 3: Java forEach() example to iterate over Stream

In this example, we are printing all the even numbers from a stream of numbers.

```
Java 8 forEach over stream elements

List<Integer> numberList = Arrays.asList(1,2,3,4,5);

Consumer<Integer> action = System.out::println;

numberList.stream()
  .filter(n -> n%2  == 0)
  .forEach( action );
```

Program output.

```
Console
```

```
2
4
```

Example 4: Java forEachOrdered() example to iterate over Stream

In this example, we are printing all the even numbers from a stream of numbers.

```
Java 8 forEachOrdered over stream elements

List<Integer> numberList = Arrays.asList(1,2,3,4,5);

Consumer<Integer> action = System.out::println;

numberList.stream()
  .filter(n -> n%2  == 0)
  .parallel()
  .forEachOrdered( action );
```

Program output.

```
Console

2
4
```

Happy Learning !!

Recommended Reading:

1. Java Stream forEach()
2. ArrayList forEach() example – Java 8
3. Java 8 – Date and Time Examples
4. Java Predicates
5. Java String join (CSV) example
6. Java Exact Arithmetic Operations Support in Math Class
7. Java 8 Optionals : Complete Reference
8. Using 'if-else' Conditions with Java Streams
9. Java Stream sorted()
0. Java Stream findFirst()

Join 7000+ Awesome Developers

Get the latest updates from industry, awesome resources, blog updates and much more.

Email Address

Subscribe

** We do not spam !!*

5 thoughts on “Java 8 forEach()”

Gaurav Pathak

February 24, 2020 at 12:59 pm

as i can see on this page :
“1. Java 8 forEach method
Below code snippet shows the default implementation of java forEach method in Iterable interface. It makes this method available to all collection classes except Map.”

it is written here that forEach() method is not available in Map interface is Wrong , please check and update the information , because as i can see in my “public interface Map” there is a “default void forEach(BiConsumer action)” method is available @since java 8.

correct me if i am wrong.

Reply

Lokesh Gupta

February 25, 2020 at 10:59 pm

Thanks Gaurav for sharing. I had missed to cover this method. The post is updated now.

Reply

Gaurav Pathak

April 18, 2020 at 12:33 pm

Hello Lokesh , the post is not updated yet. please go to the section “1.1. Iterable.forEach()” and check it is still showing that forEach() method available to all collection classes “except” Map. whereas this method is available for Map also.

If I’m incorrect/mistaken, I apologize.

Reply

Lokesh Gupta

April 18, 2020 at 3:11 pm

Hi Gaurav, Please refer to section 1.2 which talks about `forEach()` in `Map`. Section 1.1 talks about `Iterable.forEach()` which is not implemented by `Map`. Both are different.

Reply

Javeed Ahmed Sayed

August 19, 2019 at 2:41 pm

Thank you for doing this. I have been searching a lot for this 🙂

Reply

Leave a Comment

Name *

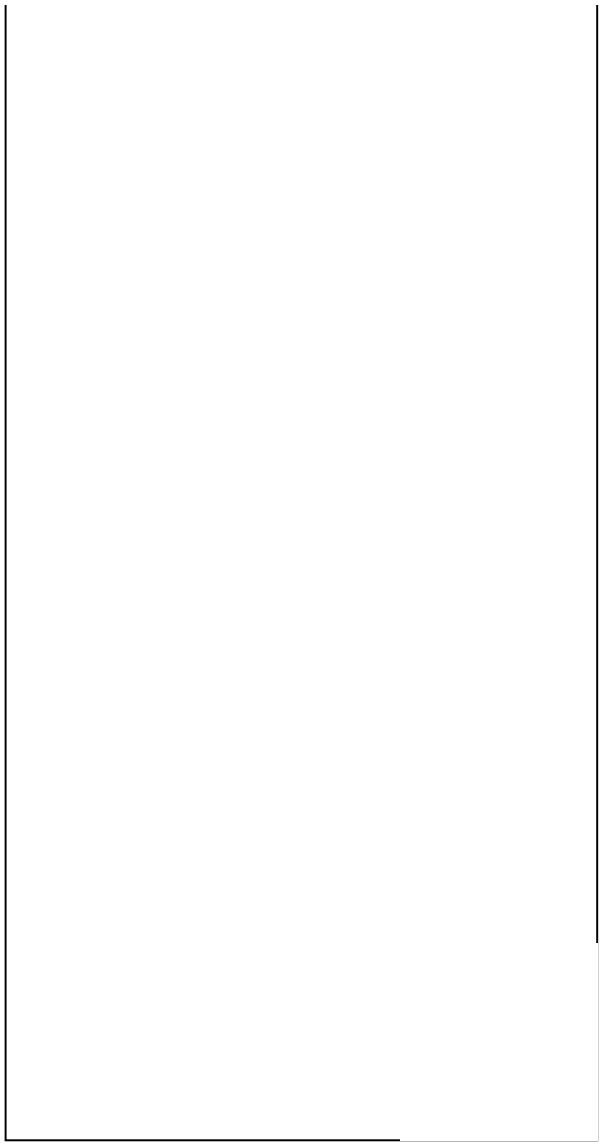
Email *

Website

Add me to your newsletter and keep me updated whenever you publish new blog posts

Post Comment

Search ...



HowToDoInJava

A blog about Java and related technologies, the best practices, algorithms, and interview questions.

Meta Links

- About Me
- Contact Us
- Privacy policy
- Advertise
- Guest Posts

Blogs

REST API Tutorial