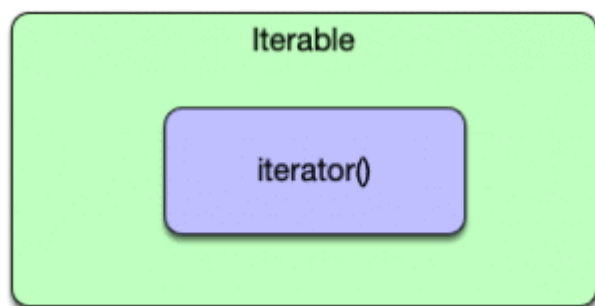


Tabla de Contenidos

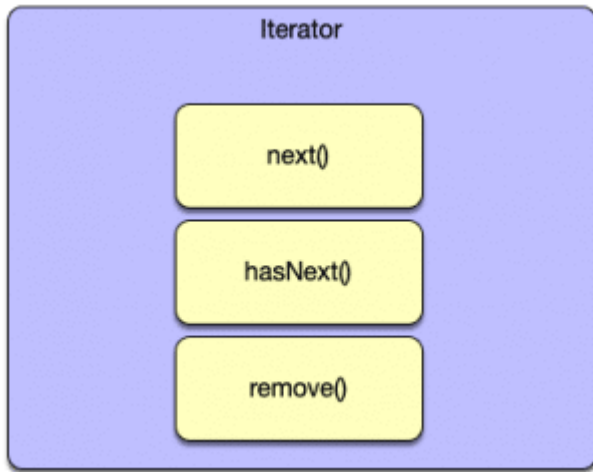


- [Java Iterable to List](#)
- [Java forEach](#)
- [Otros artículos relacionados](#)

Java Iterable to List o cómo convertir un Iterable en una lista de elementos a los cuales se puede acceder por posición . ¿Para que sirve el interface Iterable y cuál es su relación con el Interface List? . El interface Iterable es el interface más genérico del framework de colecciones y vale como su nombre indica para recorrer cualquier colección de elementos . Este interface siempre ha dispuesto de un método principal `iterator()`.



Este método nos devuelve un `Iterator` que es un interface que nos permite recorrer de forma sencilla un grupo de elementos y dispone de los métodos `hasNext()` y `next()` como principales.



Con estos métodos podemos construir cualquier colección de elementos y recorrerla:

```
Set<String> conjunto = Set.of("hola", "que", "tal");
```

```
    Iterator<String> it = conjunto.iterator();
```

```
    while (it.hasNext()) {
```

```
        System.out.println(it.next());
```

```
    }
```

El resultado lo veremos por la consola para un Set.

A screenshot of a Java console window. The title bar reads "<terminated> Principal1 [Java Application] /Library/Java/JavaVirtualMachines/jdk-14.0.1". The console output shows the words "hola", "que", and "tal" on three separate lines, with a small cursor visible at the end of the third line.

Podemos hacer la misma operación con otro tipo de colección en este caso una lista.

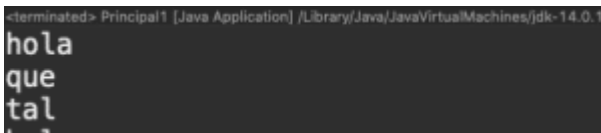
```
List<String> lista2 = List.of("hola", "que", "tal");
```

```
Iterator<String> it2 = lista2.iterator();

while (it2.hasNext()) {

    System.out.println(it2.next());
}
```

El resultado será el mismo:



```
<terminated> Principal1 [Java Application] /Library/Java/JavaVirtualMachines/jdk-14.0.1
hola
que
tal
```

Java Iterable to List

Ahora bien existen muchas situaciones en las que queremos convertir un iterable en una lista . Esto nos puede pasar tanto en código standard como en pruebas unitarias ya que un iterable solo se puede recorrer no acceder por posición . ¿Como podemos hacer esto de forma directa?. A partir de Java 8 el interface Iterable soporta el método `forEach` . De ahí que a partir de ahora sea mucho más sencillo recorrer cualquier colección .

Java forEach

Recordemos que desde Java 5 no hace falta un iterador sino que se puede usar un bucle `forEach` :

```
package com.arquitecturajava;

import java.util.List;
import java.util.Set;

public class Principal3 {
```

```

public static void main(String[] args) {

    Set<String> conjunto = Set.of("hola", "que", "tal");

    for (String cadena : conjunto) {

        System.out.println(cadena);
    }

    List<String> lista2 = List.of("hola", "que", "tal");

    for (String cadena : lista2) {

        System.out.println(cadena);
    }
}

```

A partir de Java 8 las cosas mejoran ya que el propio interface Iterable dispone de un método `forEach` que le permite de forma mucho más directa pasar una **función consumidora** y consumir la lista de forma limpia:

```

package com.arquitecturajava;

import java.util.List;
import java.util.Set;

public class Principall {

    public static void main(String[] args) {

```

```

Set<String> conjunto = Set.of("hola", "que", "tal");

conjunto.forEach(System.out::println);

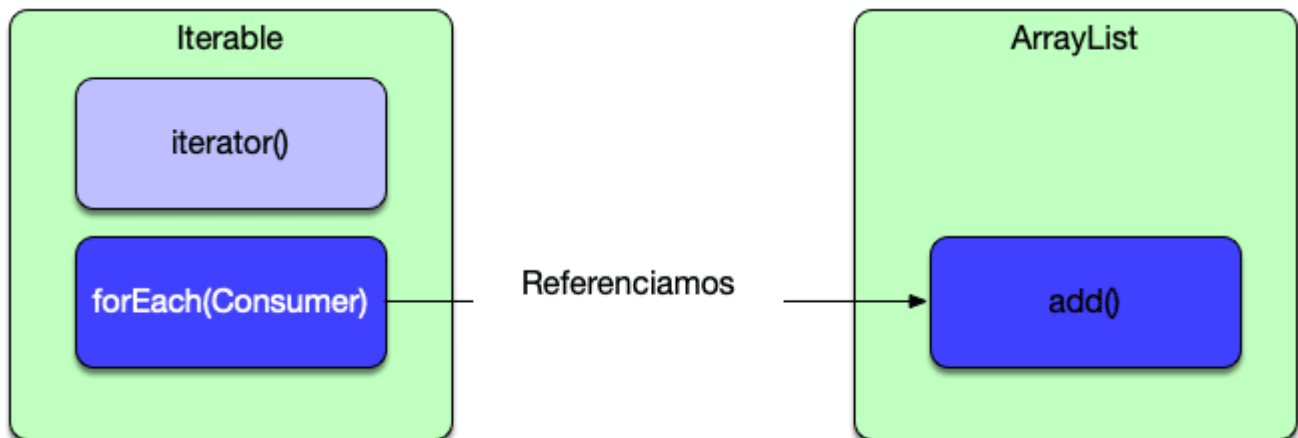
List<String> lista2 = List.of("hola", "que", "tal");

lista2.forEach(System.out::println);
}

}

```

Mucho más sencillo claramente . Es a partir de este método como podemos conseguir convertir un iterable en una lista (Iterable to List) ya que al admitir un consumidor podemos pasarle el método add del interface List.



```

package com.arquitecturajava;

import java.util.ArrayList;
import java.util.List;
import java.util.Set;

```

```
public class Principal4 {  
  
    public static void main(String[] args) {  
  
        Iterable<String> conjunto = Set.of("hola", "que", "tal");  
        List<String> nuevaLista= new ArrayList();  
        conjunto.forEach(nuevaLista::add);  
        for (int i=0;i<nuevaLista.size();i++) {  
            System.out.println(nuevaLista.get(i));  
        }  
    }  
  
}
```

Aprendamos a usar los nuevos métodos de los que disponemos en el framework de Java Collections:

Otros artículos relacionados

1. [¿Que es un Java Stream?](#)
2. [Java Functional Interface](#)
3. [Java Pattern Predicate y filtrados](#)