

Blog sobre Java EE

Estás aquí: [Inicio](#)/[Java SE](#)/[Java 8](#)/Java 8 Lambda Syntax ,simplificando nuestro código

Java 8 Lambda Syntax ,simplificando nuestro código

2 marzo, 2018 por [Cecilio Álvarez Caules](#) — [Deja un comentario](#)

Echa un vistazo a nuestra
gama de camcorders
profesionales 4K



SONY

Más información >

Las opciones soportadas por **Java 8 Lambda Syntax** siempre son costosas de aprender e integrar como parte de nuestro conocimiento Java. **¿Cuales son las sintaxis soportadas por las expresiones lambda?** .Vamos a construir un ejemplo sencillo usando un **Java Comparator**. Para ello nos vamos a construir la clase Persona y con nombre, apellidos y edad y vamos a **crear un comparador por edad con Java clásico**.

```
1 package com.arquitecturajava;
2
3 public class Persona {
4
5     private String nombre;
6     private String apellidos;
7     private int edad;
8     public String getNombre() {
9         return nombre;
10    }
11    public void setNombre(String nombre) {
12        this.nombre = nombre;
13    }
14    public String getApellidos() {
15        return apellidos;
16    }
17    public void setApellidos(String apellidos) {
18        this.apellidos = apellidos;
19    }
20    public int getEdad() {
21        return edad;
22    }
23    public void setEdad(int edad) {
24        this.edad = edad;
25    }
26    public Persona(String nombre, String apellidos, int edad) {
27        super();
28        this.nombre = nombre;
29        this.apellidos = apellidos;
```

Este sitio web utiliza cookies para que usted tenga la mejor experiencia de usuario. Si continúa navegando está dando su consentimiento para la aceptación de las mencionadas cookies y la aceptación de nuestra [política de cookies](#), pinche el enlace para mayor información.

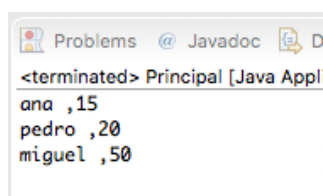
[plugin cookies](#)[ACEPTAR](#)

```
6
7     @Override
8     public int compare(Persona p1, Persona p2) {
9         // TODO Auto-generated method stub
10        return p1.getEdad()>p2.getEdad()? 1:-1;
11    }
12
13 }
```

Una vez tenemos el comparador construido el siguiente paso es construir un programa main que nos ordene la lista:

```
1 package com.arquitecturajava;
2
3 import java.util.Arrays;
4 import java.util.List;
5
6 public class Principal {
7
8     public static void main(String[] args) {
9
10
11         Persona personaA= new Persona("pedro","perez",20);
12         Persona personaB= new Persona("ana","blanco",15);
13         Persona personaC= new Persona("miguel","alvarez",50);
14
15         List<Persona> lista= Arrays.asList(personaA,personaB,personaC);
16
17         lista.sort(new PersonaEdadComparator());
18         for (Persona p: lista) {
19
20             System.out.println(p.getNombre()+" ,"+p.getEdad());
21
22
23         }
24     }
25
26 }
27
28 }
```

Imprimimos el resultado:



```
Problems @ Javadoc D
<terminated> Principal [Java Appl
ana ,15
pedro ,20
miguel ,50
```

que es posible usar una clase anónima.

Este sitio web utiliza cookies para que usted tenga la mejor experiencia de usuario. Si continúa navegando está dando su consentimiento para la aceptación de las mencionadas cookies y la aceptación de nuestra [política de cookies](#), pinche el enlace para mayor información.

[plugin cookies](#)[ACEPTAR](#)

```
3  import java.util.Arrays;
4  import java.util.Comparator;
5  import java.util.List;
6
7  public class Principal2 {
8
9      public static void main(String[] args) {
10
11          Persona personaA = new Persona("pedro", "perez", 20);
12          Persona personaB = new Persona("ana", "blanco", 15);
13          Persona personaC = new Persona("miguel", "alvarez", 50);
14
15          List<Persona> lista = Arrays.asList(personaA, personaB, personaC);
16
17          lista.sort(new Comparator<Persona>() {
18
19              @Override
20              public int compare(Persona p1, Persona p2) {
21
22                  return p1.getEdad() < p2.getEdad() ? 1 : -1;
23              }
24
25          });
26          for (Persona p : lista) {
27
28              System.out.println(p.getNombre() + " , " + p.getEdad());
29
30          }
31
32      }
33
34  }
```

El resultado en la pantalla será el mismo , **sin embargo ya no hará falta construir una clase `PersonaEdadComparator`** . Es evidente que hay ventajas sin embargo también es evidente que la sintaxis es muy compleja y no ayuda a clarificar el código. **Vamos a hacer un análisis de esta sintaxis.**

Este sitio web utiliza cookies para que usted tenga la mejor experiencia de usuario. Si continúa navegando está dando su consentimiento para la aceptación de las mencionadas cookies y la aceptación de nuestra [política de cookies](#), pinche el enlace para mayor información.

[plugin cookies](#)
[ACEPTAR](#)

}

}

Java 8 Lambda Syntax Block

La realidad es que tenemos mucho código que parece que aporta poco , sin embargo recordemos que si estamos todavía sobre el paraguas de Java 7 , **la sintaxis es obligatoria**. ¿Cómo podemos simplificarla con Java 8? . **Podemos hacer uso de expresiones lambda y eliminar gran parte del código:**

(Persona p1,Persona p2)

->

{

return

p1.getEdad() > p2.getEdad() ? 1 : -1;

}

En este caso lo que hemos hecho es eliminar la clase anónima y quedarnos únicamente con lo que se denomina **un bloque lambda** el cual implementa la funcionalidad. Si mostramos el código las cosas quedan mucho más sencillas.

```

1  package com.arquitecturajava;
2
3  import java.util.Arrays;
4  import java.util.Comparator;
5  import java.util.List;
6
7  public class Principal3 {
8
9      public static void main(String[] args) {
10
11          Persona personaA = new Persona("pedro", "perez", 20);
12          Persona personaB = new Persona("ana", "blanco", 15);
13          Persona personaC = new Persona("miguel", "alvarez", 50);
14
15          List<Persona> lista = Arrays.asList(personaA, personaB, personaC);
16

```

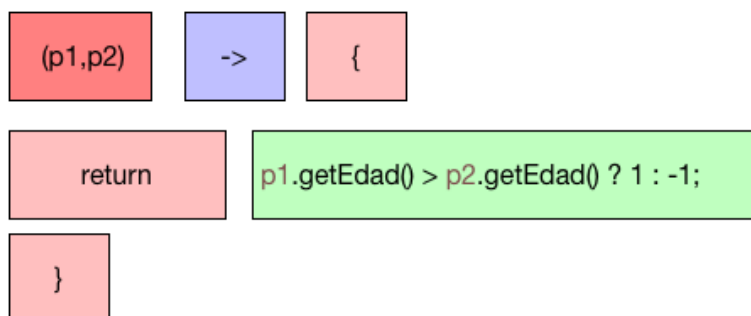
Este sitio web utiliza cookies para que usted tenga la mejor experiencia de usuario. Si continúa navegando está dando su consentimiento para la aceptación de las mencionadas cookies y la aceptación de nuestra [política de cookies](#), pinche el enlace para mayor información.

[plugin cookies](#)
[ACEPTAR](#)

```
27
28
29 }
```

Java 8 Lambda Expression

Es una simplificación importante . Ahora bien las expresiones lamdba soportan más posibilidades , una de ellas es lo que denomina **inferred types** es decir el compilador es capaz de entender de que **tipo son las variables que el método sort necesita** ya que estamos trabajando con una lista genérica de Personas. Así pues podemos simplificar todavía un poco más.



Veamos el código:

```
1 package com.arquitecturajava;
2
3 import java.util.Arrays;
4 import java.util.Comparator;
5 import java.util.List;
6
7 public class Principal4 {
8
9     public static void main(String[] args) {
10
11         Persona personaA = new Persona("pedro", "perez", 20);
12         Persona personaB = new Persona("ana", "blanco", 15);
13         Persona personaC = new Persona("miguel", "alvarez", 50);
14
15         List<Persona> lista = Arrays.asList(personaA, personaB, personaC);
16
17         lista.sort(( p1, p2)-> {return p1.getEdad() > p2.getEdad() ? 1 : -1
18         for (Persona p : lista) {
19
20             System.out.println(p.getNombre() + " , " + p.getEdad());
21
22         }
23
24     }
25 }
```

Tenemos optimizado el código bastante , pero todavía no es suficiente , estamos ante lo que se

Este sitio web utiliza cookies para que usted tenga la mejor experiencia de usuario. Si continúa navegando está dando su consentimiento para la aceptación de las mencionadas cookies y la aceptación de nuestra [política de cookies](#), pinche el enlace para mayor información.

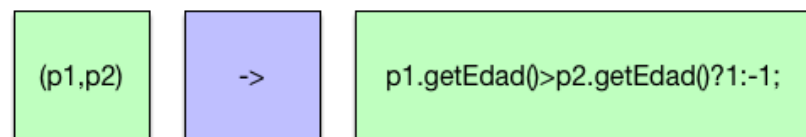
[plugin cookies](#)
[ACEPTAR](#)


Esta sintaxis ya más directa veamosla en código:

```

1  import java.util.Arrays;
2  import java.util.Comparator;
3  import java.util.List;
4
5  public class Principal4 {
6
7      public static void main(String[] args) {
8
9          Persona personaA = new Persona("pedro", "perez", 20);
10         Persona personaB = new Persona("ana", "blanco", 15);
11         Persona personaC = new Persona("miguel", "alvarez", 50);
12
13         List<Persona> lista = Arrays.asList(personaA, personaB, personaC);
14
15         lista.sort(( p1, p2)-> {return p1.getEdad() > p2.getEdad() ? 1 : -1
16         for (Persona p : lista) {
17
18             System.out.println(p.getNombre() + " , " + p.getEdad());
19
20         }
21     }
22 }
23
24 }
```

En ella podemos dar un paso adicional más ya que el compilador es capaz de entender que al ser en una única línea se pueden **eliminar los paréntesis y el return**.



Ahora nos hemos quedado con la sintaxis más compacta.

```
1 | lista.sort(( p1, p2)->p1.getEdad() > p2.getEdad() ? 1 : -1);
```

El uso de Java 8 lambda Syntax , es cada día más importante para conseguir un código más compacto y mas sencillo de entender.Tenemos poco a poco que ir eliminando el uso de clases

Este sitio web utiliza cookies para que usted tenga la mejor experiencia de usuario. Si continúa navegando está dando su consentimiento para la aceptación de las mencionadas cookies y la aceptación de nuestra [política de cookies](#), pinche el enlace para mayor información.

[plugin cookies](#)[ACEPTAR](#)

Links Externos:

Expresiones Lambda



Are your binaries fine wine or expired milk?

SCAN & SEE >



Archivada en: [Java 8](#), [Java Lambda y Streams](#)

Leave a Reply

Be the First to Comment!



Start the discussion

☒ Subscribe ▼

BUSCAR

Buscar en este sitio ...

Mis Cursos de Java Gratuitos

[Java Herencia](#)

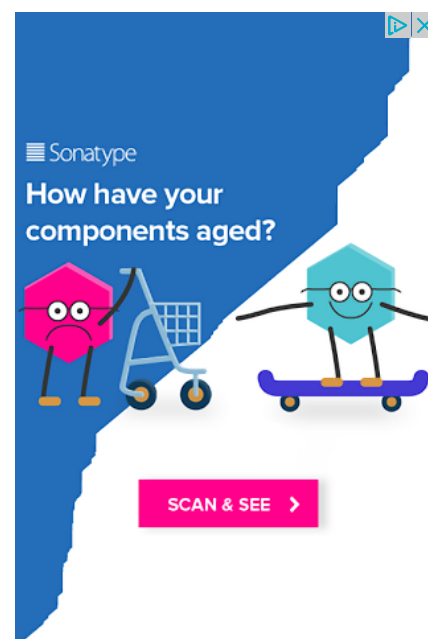
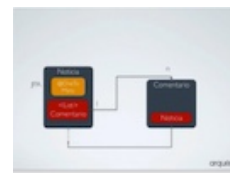


[Java JDBC](#)

Este sitio web utiliza cookies para que usted tenga la mejor experiencia de usuario. Si continúa navegando está dando su consentimiento para la aceptación de las mencionadas cookies y la aceptación de nuestra [política de cookies](#), pinche el enlace para mayor información.

[plugin cookies](#)[ACEPTAR](#)

Introduccion JPA



POPULAR

[Maven Parent POM y uso de librerías](#)[Java Generic Repository y JPA](#)[Spring REST Client con RestTemplates](#)[Spring GetMapping ,PostMapping etc](#)

[POSTMAN API CON SPRING REST](#)

Este sitio web utiliza cookies para que usted tenga la mejor experiencia de usuario. Si continúa navegando está dando su consentimiento para la aceptación de las mencionadas cookies y la aceptación de nuestra [política de cookies](#), pinche el enlace para mayor información.

[plugin cookies](#)[ACEPTAR](#)[Nuevo Curso:Arquitectura Java Sólida con Spring 4.3 y Anotaciones](#)[Spring Boot Properties utilizando @Value](#)

CONTACTO

contacto@arquitecturajava.com

LO MAS LEIDO

[¿Qué es Spring Boot?](#)[Java 8 Lambda Syntax ,simplificando nuestro código](#)[Java Constructores this\(\) y super\(\)](#)[Usando Java Session en aplicaciones web](#)[Nuevo Curso:Arquitectura Java Sólida con Spring 4.3 y Anotaciones](#)[Spring @Qualifier utilizando @Autowired](#)[Java Iterator vs ForEach](#)[¿Cuales son las certificaciones Java?](#)[Introducción a Servicios REST](#)[¿Qué es Gradle?](#)[Ejemplo de JPA , Introducción \(I\)](#)[Ejemplo de Java Singleton \(Patrones y ClassLoaders\)](#)[REST JSON y Java](#)[Usando el patron factory](#)[Java Override y encapsulación](#)[Angular 5 Hello World y su funcionamiento](#)[Mis Libros](#)[Uso de Java Generics \(I\)](#)

Este sitio web utiliza cookies para que usted tenga la mejor experiencia de usuario. Si continúa navegando está dando su consentimiento para la aceptación de las mencionadas cookies y la aceptación de nuestra [política de cookies](#), pinche el enlace para mayor información.

[plugin cookies](#)[ACEPTAR](#)

Copyright © 2018 · [eleven40 Pro Theme](#) en [Genesis Framework](#) · [WordPress](#) · [Acceder](#)