

Blog sobre Java EE

Estás aquí: [Inicio](#)/[Spring](#)/[Spring Avanzado](#)/Spring Boot @Lazy Components

Spring Boot @Lazy Components

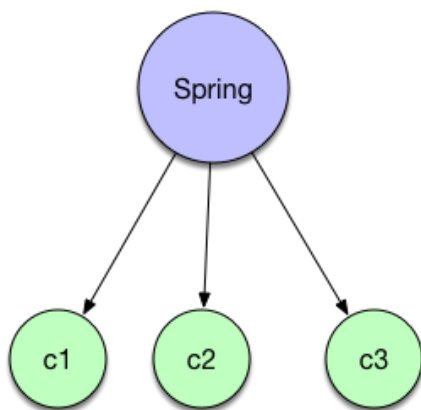
20 febrero, 2018 por [Cecilio Álvarez Caules](#) — [Deja un comentario](#)

TIBCO Jaspersoft Oficial - Presentando Jaspersoft 6

Informes y Dashboards Interactivos Embeber fácilmente en cualquier app [jaspersoft.com](#)



El uso de **Spring Boot @Lazy**, nos puede ser útil cuando construimos aplicaciones **de gran tamaño** y tenemos componentes en Spring Framework que por su funcionalidad **tardan en cargar**. Cuando Spring Framework se inicializa, por defecto genera una instancia de cada uno de sus componentes en memoria.



Spring Boot y componentes

Sin embargo puede haber situaciones en las que deseemos una carga vaga (lazy) de alguno de estos componentes.

[JAVA SE](#) [JAVA SE +](#) [SPRING](#) [JAVA EE](#) [JAVASCRIPT](#) [FRAMEWORKS JS](#) [ARQUITECTURA](#) [MIS LIBROS](#)

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](#)

Vamos a ver un ejemplo sencillo de como utilizar este enfoque. Supongamos que tenemos los siguientes interfaces y clases definidos con una aplicación de Spring Boot.

```
1 package com.arquitecturajava.lazy;
2
3 public interface ServicioA {
4     String mensajeA();
5 }
6
7
8 package com.arquitecturajava.lazy;
9
10 import org.springframework.stereotype.Service;
11
12 @Service
13 public class ServicioAImpl implements ServicioA {
14
15     public ServicioAImpl() {
16         System.out.println("instancia ServicioA");
17     }
18
19     @Override
20     public String mensajeA() {
21         return "hola servicioA";
22     }
23 }
24
25 package com.arquitecturajava.lazy;
26
27 public interface ServicioB {
28     String mensajeB();
29 }
30
31
32 package com.arquitecturajava.lazy;
33
34 import org.springframework.stereotype.Service;
35
36 @Service
37 public class ServicioBImpl implements ServicioB {
```

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](#)

```

17     return "hola servicioC";
18     }
19 }

1 package com.arquitecturajava.lazy;
2
3 public interface ServicioC {
4     String mensajeC();
5 }
6
7
8
9
10 package com.arquitecturajava.lazy;
11
12 import org.springframework.stereotype.Service;
13
14 @Service
15 public class ServicioCImpl implements ServicioC {
16
17     public ServicioCImpl() {
18         System.out.println("instancia ServicioC");
19     }
20
21     @Override
22     public String mensajeC() {
23         return "hola servicioC";
24     }
25 }

```

Spring Boot estandar

En estos momentos tenemos tres Beans que se cargarán con Spring Boot al arrancar vamos a ver como queda el programa principal de la consola para ejecutar **los métodos de estos beans**.

```

1 package com.arquitecturajava.lazy;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4 import org.springframework.boot.CommandLineRunner;
5 import org.springframework.boot.SpringApplication;
6 import org.springframework.boot.autoconfigure.SpringBootApplication;
7
8 @SpringBootApplication
9 public class LazyApplication implements CommandLineRunner {
10
11     @Autowired
12     ServicioA miservicioA;
13     @Autowired
14     ServicioB miservicioB;

```

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.

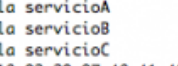
plugin cookies

ACCEPTAR

```
25  
26         System.out.println(miservicioA.mensajeA());  
27         System.out.println(miservicioB.mensajeB());  
28         System.out.println(miservicioC.mensajeC());  
29     }  
30 }  
31 }
```

Spring Boot y la consola

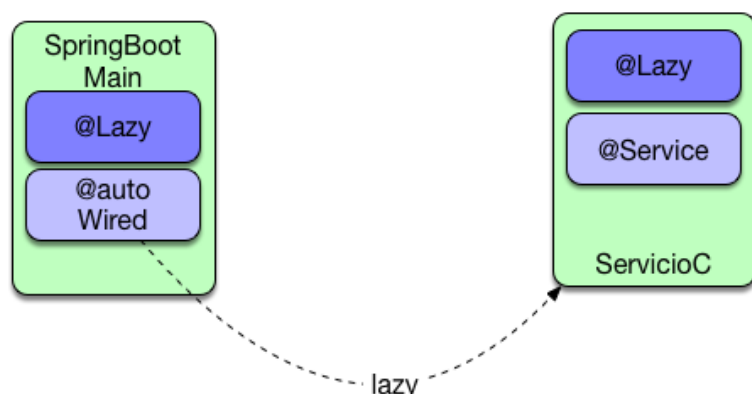
Si ejecutamos el programa veremos que los servicios se instancian todos a la vez y luego se ejecutan las invocaciones a cada uno de sus métodos.

```

:: Spring Boot ::                (v1.5.10.RELEASE)

2018-02-20 07:48:41.012 INFO 1340 --- [
2018-02-20 07:48:41.014 INFO 1340 --- [
2018-02-20 07:48:41.053 INFO 1340 --- [
instancia ServicioA
instancia ServicioB
instancia ServicioC
2018-02-20 07:48:41.471 INFO 1340 --- [
hola servicioA
hola servicioB
hola servicioC
2018-02-20 07:48:41.483 INFO 1340 --- [
2018-02-20 07:48:41.484 INFO 1340 --- [
2018-02-20 07:48:41.485 INFO 1340 --- [
```

Spring Boot @Lazy

Cómo podemos modificar el comportamiento del programa para que el servicioC se cargue de forma vaga. Es bastante sencillo bastará con marcar que el servicio se puede instanciar de forma vaga y cuando hagamos el @Autowired confirmar que la carga es de esta forma.



plugin cookies

ACCEPTAR

```
1 | @Lazy
2 | @Autowired
3 | ServicioC miservicioC;
```

[illegible]

Otros artículos relacionados:

1. Spring Boot WAR sin Microservicios
2. ¿Qué es Spring Boot?
3. Spring Boot WAR sin Microservicios
4. Spring Boot AOP y rendimiento

Enlaces Externos:

- ## 1. Spring Boot



in



1



1
COMPART

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

Leave a Reply

Be the First to Comment!

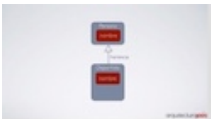
Notify of



BUSCAR

Mis Cursos de Java Gratuitos

Java Herencia



Java JDBC

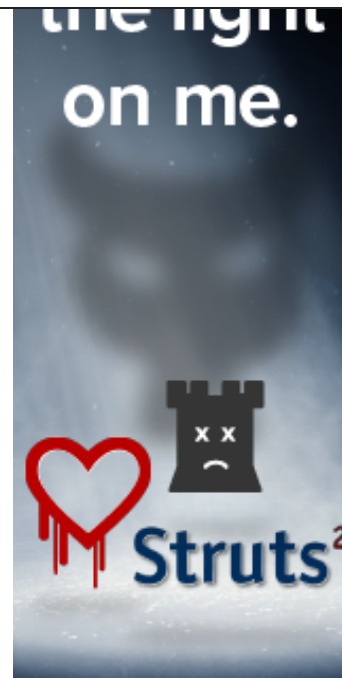


Servlets



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](#)



POPULAR

[Maven Parent POM y uso de librerías](#)[Java Generic Repository y JPA](#)[Spring GetMapping ,PostMapping etc](#)[Spring REST Client con RestTemplates](#)[Java 9 Modules y el concepto de modularidad](#)[Java Interfaces y el concepto de simplicidad](#)[PostMan App con Spring REST](#)[El concepto de Java Annotations y su funcionamiento](#)[Spring Boot Properties utilizando @Value](#)[Java 9 Collections y sus novedades](#)

CONTACTO

contacto@arquitecturajava.com

[nuevo curso: arquitectura java spring](#)

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](#)[funcionamiento](#)[Usando Java Session en aplicaciones web](#)[Java Constructores this\(\) y super\(\)](#)[Java Iterator vs ForEach](#)[Introducción a Servicios REST](#)[¿Cuales son las certificaciones Java?](#)[¿Qué es Gradle?](#)[Ejemplo de JPA , Introducción \(I\)](#)[REST JSON y Java](#)[Ejemplo de Java Singleton \(Patrones y ClassLoaders\)](#)[RxJS Ajax , fusionando peticiones asíncronas](#)[Usando el patron factory](#)[El concepto de Java Annotations y su funcionamiento](#)[Java Override y encapsulación](#)[Mis Libros](#)[Uso de Java Generics \(I\)](#)[¿Qué es un Microservicio?](#)