

## Blog sobre Java EE

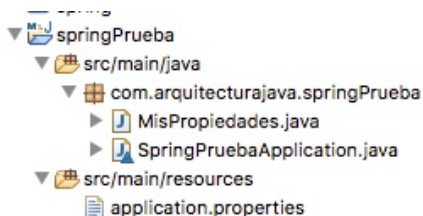
Estás aquí: [Inicio/Sin categoría/Spring Boot Properties utilizando @Value](#)

# Spring Boot Properties utilizando @Value

29 noviembre, 2017 por [Cecilio Álvarez Caules](#) — [Deja un comentario](#)



El uso de **Spring Boot properties** es muy habitual cuando trabajamos con una aplicación de Spring Boot. A diferencia de otras aplicaciones clásicas de Spring Framework, Spring Boot hace uso del principio de convención sobre configuración y define un fichero por defecto de propiedades. Este fichero se encuentra en la carpeta resources de nuestro proyecto.



## Spring Boot Properties

Vamos a usar este fichero de propiedades para dar de alta dos valores (un nombre y un apellido) y acceder a ellos desde nuestra aplicación. Tenemos que configurar nuestra aplicación de Spring Boot con las anotaciones necesarias. En nuestro caso nos vamos a definir una clase Java sencilla que contenga ambas propiedades.

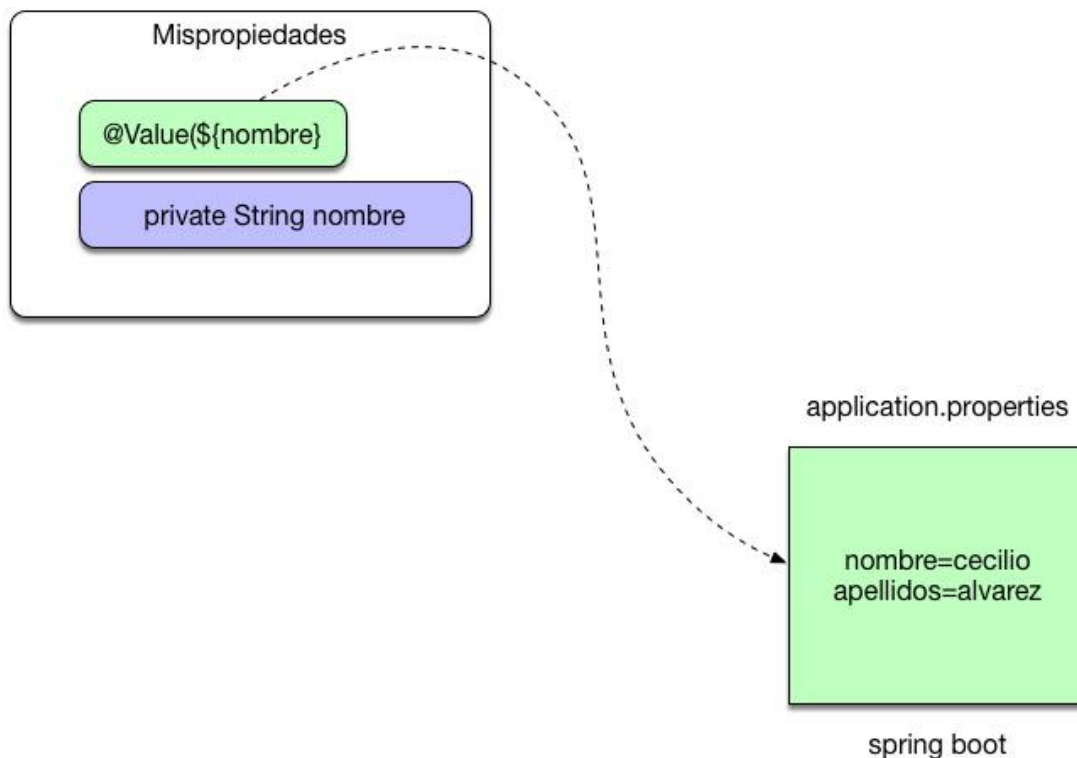
```
1 package com.arquitecturajava.springPrueba;
2
3 import org.springframework.beans.factory.annotation.Value;
4 import org.springframework.stereotype.Component;
5
6 @Component
7 public class MisPropiedades {
8     @Value("${nombre}")
9     private String nombre;
10    @Value("${apellidos}")
11    private String apellidos;
12    public String getNombre() {
13        return nombre;
14    }
15 }
```

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](#)

```
24  
25 }
```

Como podemos observar nos hemos apoyado en la anotación **@Value** para **inicializar las propiedades**.



Hemos hecho uso de spring expression language para acceder a cada uno de los valores **utilizando \${valor}**. De esta forma tan sencilla seremos ya capaces de acceder a las propiedades definidas en el fichero application.properties. Nos queda simplemente inyectar nuestro clase en el proyecto de SpringBoot y ejecutarlo **como proyecto de consola para ver el resultado**.

```
1 package com.arquitecturajava.springPrueba;
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 class SpringPruebaApplication implements CommandLineRunner {
10
11     @Autowired
```

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](#)

```
23  
24         System.out.println(propiedades.getNombre());  
25         System.out.println(propiedades.getApellidos());  
26  
27     }  
28  
29 }
```

Ahora nos es suficiente con ejecutar nuestra aplicación desde Eclipse y veremos como se imprime el nombre y los apellidos por la consola.

```
2017-11-29 10:46:42.101 INFO 40772 --- [
2017-11-29 10:46:42.104 INFO 40772 --- [
2017-11-29 10:46:42.167 INFO 40772 --- [
2017-11-29 10:46:42.777 INFO 40772 --- [
cecilio
alvarez
2017-11-29 10:46:42.798 INFO 40772 --- [
2017-11-29 10:46:42.799 INFO 40772 --- [
2017-11-29 10:46:42.800 INFO 40772 --- [
```

Acabamos de utilizar las capacidades de Spring Boot para manejo de propiedades.

Otros artículos relacionados

1. [Spring Boot WAR sin Microservicios](#)
2. [¿Qué es Spring Boot?](#)
3. [Spring Boot AOP y rendimiento](#)
4. [Spring Boot](#)



20

20  
COMPART

price drop

price drop

Archivada en: [Sin categoría](#)

Etiquetada con: [SpringCore](#)

Leave a Reply

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](#)

## 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](#)

Almacenamiento en la  
nube seguro, ilin  
costeable y esc  
para toda func



## POPULAR

[Maven Parent POM y uso de librerías](#)[Mis Cursos de Java para desarrolladores](#)[Spring GetMapping ,PostMapping etc](#)[Java Stream map y estadísticas](#)[Java Generic Repository y JPA](#)[El concepto de JavaScript Proxy](#)[Spring 5 Hello World](#)[Spring Boot WAR sin Microservicios](#)[Java Stream Context y simplificación de Streams](#)[Java Stream File y manejo de ficheros](#)

## CONTACTO

[contacto@arquitecturajava.com](mailto:contacto@arquitecturajava.com)

## LO MAS LEIDO

---

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](#)

---

[Introducción a Servicios REST](#)

---

[¿Cuales son las certificaciones Java?](#)

---

[Ejemplo de JPA , Introducción \(I\)](#)

---

[¿Qué es Gradle?](#)

---

[Java Generic Repository y JPA](#)

---

[Ejemplo de Java Singleton \(Patrones y ClassLoaders\)](#)

---

[Usando el patron factory](#)

---

[REST JSON y Java](#)

---

[Java Stream File y manejo de ficheros](#)

---

[Uso de Java Generics \(I\)](#)

---

[Mis Libros](#)

---

[Comparando java == vs equals](#)

---

[Black Friday y cupones de descuento del 50% :\)](#)

---

[Java Override y encapsulación](#)

---

[Spring MVC Configuración \(I\)](#)

---