# Punto 1: El Laboratorio de los Beans

**Elaborado por:** Andrés Felipe Muzu Yupangui Tisoy

## Ciclo de vida y diferencias

El ciclo de vida de los beans comienza con su definición, ya sea mediante anotaciones automáticas como @Component o de forma manual con @Bean en una clase de configuración. Cuando la aplicación inicia, los beans se instancian y sus constructores se ejecutan, lo que permite observar en consola el momento exacto en que cada uno es creado. El uso de @Component facilita la detección automática de clases propias, mientras que @Bean otorga mayor control y flexibilidad para instanciar objetos de terceros o con lógica personalizada.

## Reto 1 y Reto 2

En estos retos se evidencia cómo ambos beans, ExperimentService anotado con @Component y ManualService creado manualmente con @Bean, se instancian durante el arranque de la aplicación antes de que se ejecute cualquier lógica de negocio, mostrando en consola sus mensajes de construcción de forma inmediata al iniciar el contexto de Spring. Si no se utiliza la anotación @Lazy, este comportamiento de inicialización temprana ocurre siempre, creando todos los beans durante el startup de la aplicación. 

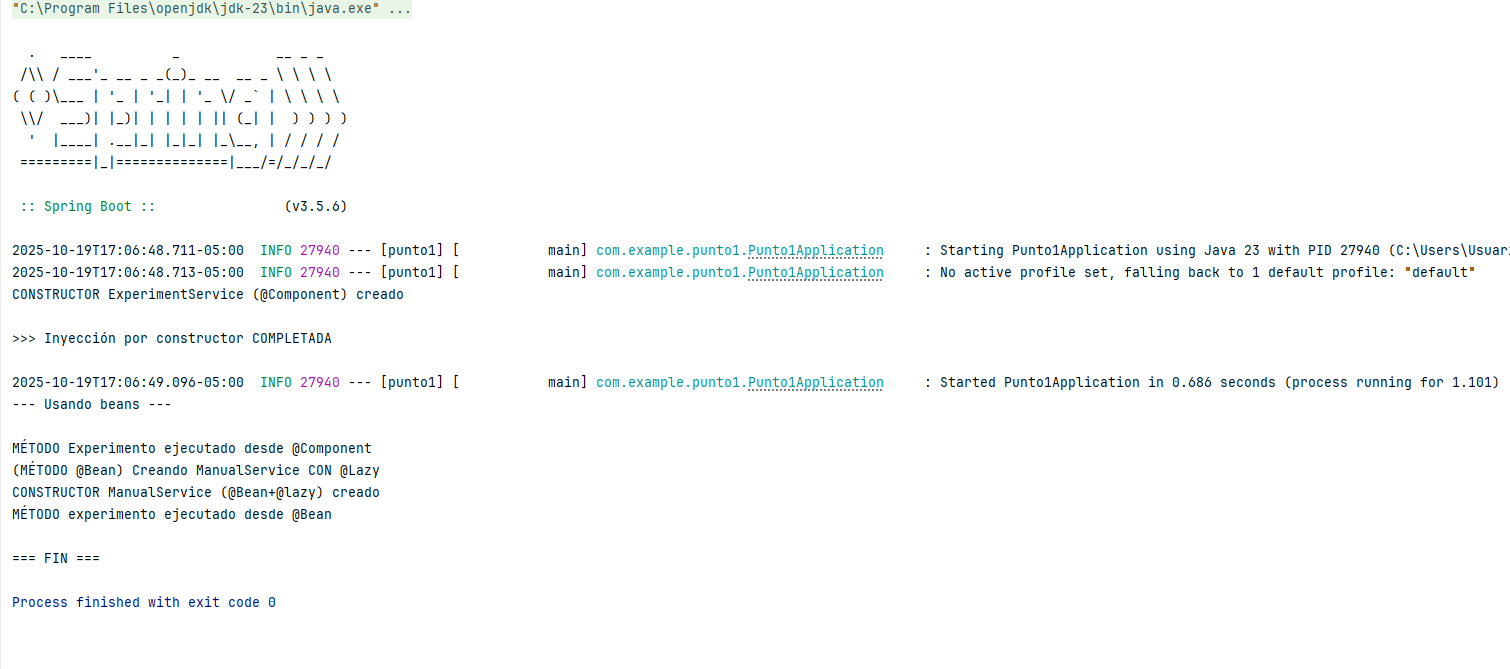
## Reto 3

En este reto se demuestra que al aplicar @Lazy tanto en la definición de ExperimentService como en el punto de inyección del constructor, Spring inyecta un proxy en vez del bean real y retrasa la creación del objeto hasta que se utiliza por primera vez en el método run(), permitiendo que solo ManualService se cree durante el startup mientras ExperimentService permanece sin instanciar hasta su primer uso. Esta diferencia es clave para optimizar recursos y tiempos de inicio, ya que permite que los beans costosos solo se creen cuando realmente son necesarios.



## Reto 4

De manera complementaria, en este reto se intercambia la configuración aplicando @Lazy únicamente a ManualService, evidenciando que ahora ExperimentService se crea de forma inmediata durante el arranque mientras ManualService se inicializa cuando se invoca su método por primera vez, confirmando que el comportamiento de inicialización lazy es independiente de si el bean se define con @Component o @Bean.



## Notas adicionales:

No se si será de mucha importancia, pero se tuvo un problema que era debería ser bastante sencillo de solucionar, pero pase más tiempo del que me gustaría admitir dándole vueltas, es de esa clase de errores que puede pasar desapercibido en códigos mas extensos. El problema radicaba en que la anotación @Lazy solo estaba presente en la definición del bean, pero no en el punto de inyección dentro del constructor de Punto1Application. Cuando Spring intenta inyectar dependencias mediante el constructor, necesita proporcionar una instancia real del bean para poder ejecutar el constructor, lo que obliga al contenedor a crear el bean inmediatamente, anulando por completo el efecto de @Lazy.