educación IT
CODE YOUR FUTURE

Bootcamp Java Developer

Fase 3 - Java Architect

Módulo 21

educación IT
CODE YOUR FUTURE

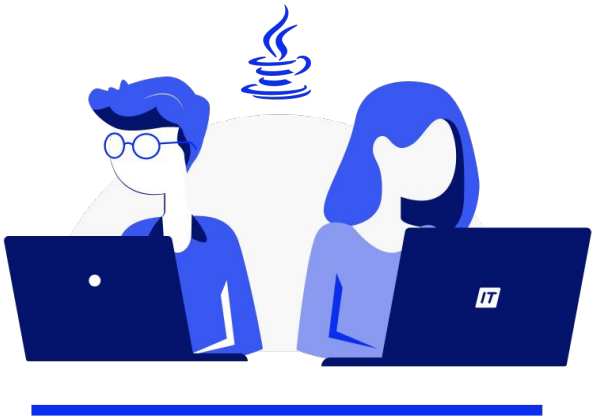
Spring Beans

¿Qué es un bean en Java?

Es un componente de *software* que tiene la particularidad de ser **reutilizable.**

En el lenguaje de programación Java, cumple varios criterios:

1. Tiene todos sus **atributos privados (`private`).**
2. Tiene **métodos `set()` y `get()` públicos** de los atributos privados que nos interesen.
3. Tienen un **constructor público** por defecto.



Beans en Spring Framework

A diferencia de los tradicionales que representan una clase, **en Spring Framework los *beans* son objetos creados y manejados por el contenedor Spring.**

El contenedor se encuentra en el núcleo del marco de trabajo y utiliza inyección de dependencias para gestionar los componentes que forman la aplicación. Se encarga de varias tareas, como crear, conectar y alojar los objetos definidos por los *beans.* Además, hace de dispensador proporcionando *beans* por petición. El contenedor se encarga de cargar las definiciones de *beans.*

Tipos de contenedor de Spring:

1. **Bean Factory:** contenedor sencillo con soporte básico de inyección de dependencias.
2. **Application Context:** es una implementación de un *bean factory* que proporciona opciones avanzadas como:
    a. Medios para resolver mensajes de texto e internalización.
    b. Publicación de beans registrados como receptores o formas genéricas de abrir recursos de archivo.

Ciclo de vida de un bean

Los *beans* de Spring Framework tienen un ciclo de vida en el contexto de la aplicación. Podemos ordenar estas fases de vida de esta manera:

Formas de crear un bean

1. **Bean simples:** no poseen atributos.
2. **Bean con inyección por constructor:** pasando los atributos por constructor.
3. **Bean con referencias de objeto de constructores:** cuando pasamos un *bean* como atributo del constructor de otro.
4. **Bean con inyección de propiedades:** cuando en vez del método constructor utilizamos métodos `setters` de atributos.
5. **Con valores simples:** enteros, reales, y otros.
6. **Con valores complejos:**
    a. **Por referencia de otro objeto:** Pasando un `bean id` al método `set`.
    b. **Colecciones de datos:** listas, *arrays, maps.*
    c. **Con valor nulo:** cuando necesitamos pasar un valor nulo.

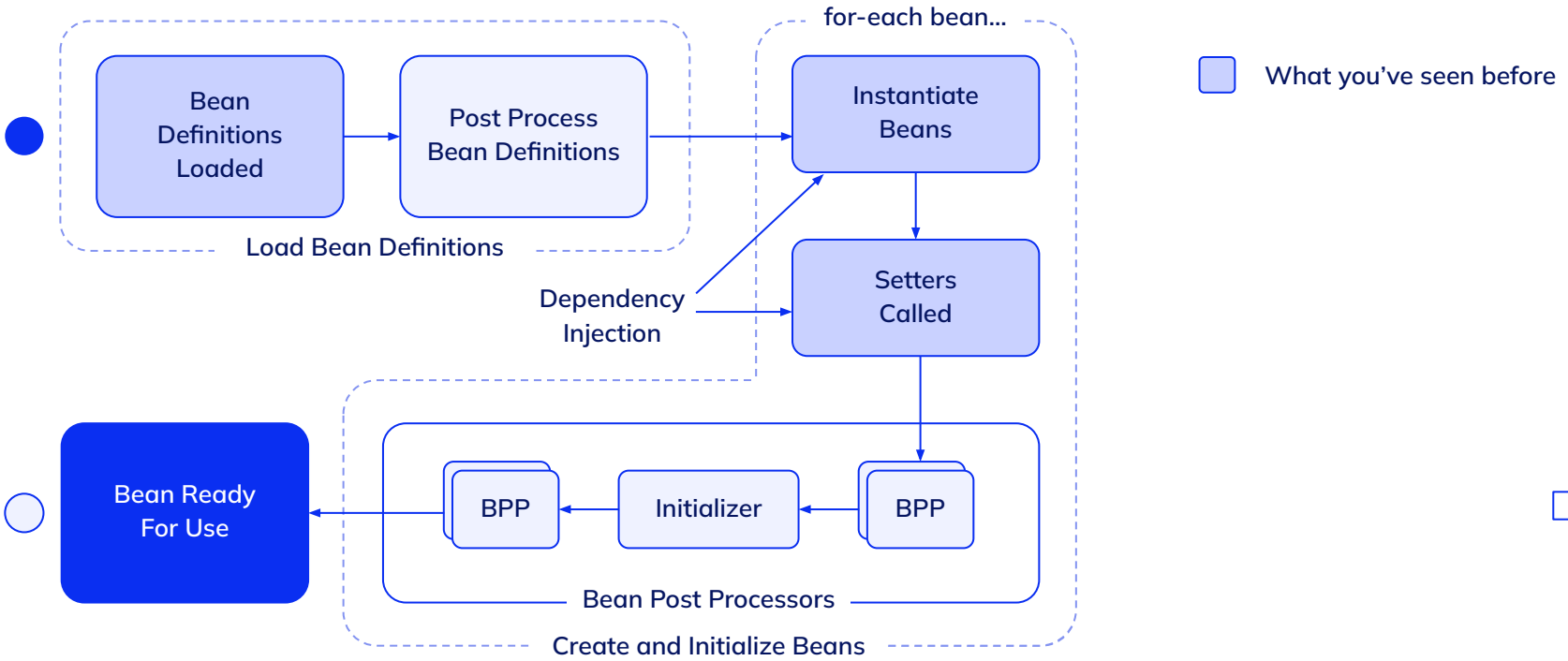
Ámbito de un bean

Cuando trabajamos con un *bean* en Spring Framework por defecto son **`singletons`**. ¿Qué significa esto? Significa que **el contenedor solo instancia un objeto de la clase**, y cada vez que se pide una instancia del *bean* en realidad se obtiene una referencia al mismo objeto.

El ámbito `singleton` es el indicado en muchos casos, pero si necesitamos cambiar este comportamiento, **podemos asignar otros ámbitos para el *bean*** como:

- Para especificar que queremos una nueva instancia cada vez que se solicite el *bean*, se usa el valor **`prototype`**.
- En aplicaciones web, se pueden usar los ámbitos de **`request`** y **`session`**.

Pasos de inicialización del bean



Aplicaciones más comunes utilizadas durante el tiempo de desarrollo en Spring Framework:

`@Bean`	Define un *bean* dentro del *application context.*
`@Scope`	Indica el ámbito del *bean.*
`@Service`	Indica que el *bean* creado es un posible servicio.
`@Component`	Indica que el *bean* creado es un posible componente.
`@Repository`	Indica que el *bean* creado es un objeto de acceso a datos.
`@Controller`	Indica que el *bean* creado es un componente Web.
`@PostConstruct`	Indica que el método en el *bean* será disparado luego de ser llamado el constructor de la clase.
`@PreDestroy`	Indica que el método en el *bean* será disparado luego de ser eliminado el *bean* del contexto.
`@Autowired`	Indica que el contexto debe inicializar el argumento marcado en el *bean.*

Anotaciones

Las anotaciones en Java son formas de añadir metadatos al código fuente para que estén disponibles para la aplicación en tiempo de ejecución. Las anotaciones Java pueden añadirse a los elementos de programa tales como clases, métodos, campos, parámetros, variables locales, y paquetes.

¡Sigamos trabajando!